Titre: Title:	Techniques d'optimisation de la navigation globale d'un fauteuil roulant motorisé intelligent
Auteur: Author:	Ghazi Majdoub
Date:	2014
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Majdoub, G. (2014). Techniques d'optimisation de la navigation globale d'un fauteuil roulant motorisé intelligent [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1482/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/1482/
Directeurs de recherche: Advisors:	Richard Gourdeau, & Joëlle Pineau
Programme: Program:	génie électrique

UNIVERSITÉ DE MONTRÉAL

TECHNIQUES D'OPTIMISATION DE LA NAVIGATION GLOBALE D'UN FAUTEUIL ROULANT MOTORISÉ INTELLIGENT

GHAZI MAJDOUB
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
AOÛT 2014

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

TECHNIQUES D'OPTIMISATION DE LA NAVIGATION GLOBALE D'UN FAUTEUIL ROULANT MOTORISÉ INTELLIGENT

présenté par : MAJDOUB Ghazi
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. SAUSSIÉ David, Ph.D., président
M. GOURDEAU Richard, Ph.D., membre et directeur de recherche
Mme PINEAU Joelle, Ph.D., membre et codirectrice de recherche
M. LE NY Jérôme, Ph.D., membre

À tous mes frères et sœurs,
je vous aime...

REMERCIEMENTS

Ce projet aurait difficilement abouti sans les personnes suivantes.

Mes directeurs de recherche, Richard Gourdeau et Joelle Pineau, qui m'ont proposé un sujet de maîtrise intéressant et extrêmement enrichissant. Je les remercie de m'avoir guidé et soutenu durant ce projet. J'ai bien apprécié la recherche sous leur direction marquée par l'ouverture et l'équilibre entre l'encadrement et l'autonomie.

Je remercie Hai Nguyen avec qui j'ai passé le plus de temps en laboratoire. Son dévouement et sa serviabilité m'inpressionnent. C'est grâce à lui que j'ai pu m'affronchir des aspects matériels du système.

Je remercie tout les professeurs qui m'ont enseigné à l'École Polytechnique. En particulier, je remercie Jérôme le Ny, enseignant du cours de Navigation Aérienne, qui m'a offert l'occasion de démarrer ce projet sur de bonnes bases dans le cadre du mini-projet du cours. Je remercie également tout mes amis, surtout mes amis Karim Bessadi et Wassim Rafrafi. Ils étaient toujours présents pour m'éclairer par leurs conseils précieux. Nos discussions m'étaient une source d'inspiration et d'encouragement. Ils me sont tels de vrais frères.

Je remercie mon responsable de recherche à SupAéro David Mimoun, ainsi que les responsables des échanges internationaux Françoise Loytier et Marie-Paul Lozano.

Je remercie tout les organismes qui ont soutenu ce projet : *NSERC Canadian Network on Field Robotics*, Regroupements stratégiques *REPARTI* et *INTER* (financé par *FQRNT*), *CanWheel* (financé par *CIHR*), *CRIR Rehabilitation Living Lab* (financé par *FRQS*), et *NSERC Discovery grant program.*

Je remercie énormément mes parents, ma femme et mon premier fils né à la fin de ce projet. A tout mes frères et sœurs, c'est votre amour qui me motive et me remplit d'énérgie. Je remercie Dieu de m'avoir comblé de sa générosité. Sans son amour et sa bienveillance, je n'aurais pas pu réaliser tout ce que j'ai réalisé.

RÉSUMÉ

Un projet de réalisation d'un prototype de fauteuil roulant motorisé intelligent (FRMI) avait réuni depuis quelques années les efforts de chercheurs et d'étudiants en cycle supérieur de troix universités du Québec : École Polytechnique, Université de McGill et Université de Montréal. Ces efforts ont abouti à un fauteuil équipé de capteurs et d'un ensemble de modules permettant l'automatisation de nombreuses tâches. Il est ainsi un cas particulier de robot mobile.

Malgré l'état avancé du fauteuil, plusieurs perspectives d'améliorations restent ouvertes. En effet, les modules de navigation globale existants se basent sur un algorithme de cartographie et localisation simultanées qui n'utilise qu'un modèle de perception adapté aux seules données des télémètres laser et qui présente quelques défauts dont la faible performance des capteurs utilisés pour certains types d'obstacles : les vitres et les objets transparents notamment sont difficiles à détecter avec des capteurs optiques comme les télémètres lasers. Le premier objectif principal du projet serait alors de proposer une solution à ce problème en partant du module de SLAM utilisé, *the GMapping algorithm*. On se propose également de proposer une alternative à la stratégie de cartographie par SLAM qui impose l'exploration des environnements visités. Notre second objectif consiste alors à utiliser les plans architecturaux pour générer automatiquement des grilles d'occupation compatibles avec le système de navigation du fauteuil.

Notre réalisation de ces objectifs est exposée à travers l'étude du module à optimiser, l'analyse de ses composants et la conception des algorithmes-solutions. La première solution consiste essentiellement à intégrer un composant mieux adapté aux obstacles transparents et qui permet un enrichissement en temps réel par les détections des sonars de la carte en cours de construction par le *GMapping*. D'un autre côté, une technique toute nouvelle de construction de cartes est implémentée et testée grâce à un algorithme qui analyse les calques du plan d'architecture au format DXF afin d'extraire les éléments intéressants pour la navigation.

Les résultats d'enrichissement montrent des cartes plus complètes où les obstacles transparents figurent aux bons emplacements. Nous avons pu profiter de la capacité des sonars à détecter les objets transparents sans pour autant perdre les hautes performances des télémètres laser et sans altérer la construction en temps réel de la carte accompagnée de la localisation. D'autre part, nos grilles d'occupation générées automatiquement ont été intégrées et utilisées avec succès dans le système de navigation. Une cartographie rapide et nettement plus économique est désormais possible pour le FRMI à condition d'avoir un plan

au format DXF du milieu à modéliser.

ABSTRACT

For the last few years, researchers and students of three universities of Quebec, École Polytechnique, Université de Montréal and McGill University, have been working on the creation of an Intelligent Powered Wheelchair (IPW). Thanks to their work, a wheelchair equiped with sensors and using a wide range of modules enabling the chair to do numerous automatic tasks became a reality. The IPW can therefore be considered as an example of a mobile robot.

Many possible improvements still can be performed in spite of the very advanced state of the wheelchair. The global navigation is based on a simultaneous localization and mapping (SLAM) algorithm; *the GMapping algorithm* that uses a perception model conceved for lasers detections only and that is therefore under the required level of efficiency when mapping is held in environments containing transparent obstacles such as glass walls. Besides, the SLAM algorithm used is, as any other SLAM solution, based on a mapping strategy that imposes the environment exploration. Thus, our work's objectives are: First, to propose a solution to the transparent objects problem and second, to suggest a new alternative to the old exploration based strategy by exloiting blueprints to generate occupancy grid maps automatically.

This report shows how we achieve these objectives through the study and analysis of the available SLAM algorithm and the conception of our solutions. In the first solution, we introduce a component that takes sonars detections as inputs and that draws in real time, based on a sonar perception model, the missing transparent objects on the *GMapping* map. In our second solution, a new technique makes use of an algorithm that analyses the layers of CAD blueprints given in a DXF file format in order to extract the important elements used for grids construction.

Our algorithms proved to be efficient. In fact, we managed to get complete maps showing the transparent obstacles at the right spots. We managed then to make use of the sonar's capability to see transparent objects without causing the decrease of the *GMapping* performance in both mapping with laser data and localisation. On the other hand, we managed to automatically build occupancy grid maps from blueprints. The built maps proved precise when used successfully in the wheeler's navigation system. We have now a faster and cheaper mapping solution for the IPW.

TABLE DES MATIÈRES

DÉDICACE iii

REMERCIEMENTS iv

RÉSUMÉ v

ABSTRACT vii

TABLE DES MATIÈRES viii

LISTE DES TABLEAUX xi

LISTE DES FIGURES xii

LISTE DES ANNEXES xv

LISTE DES SIGLES ET ABRÉVIATIONS xvi

CHAPITRE 1 INTRODUCTION 1
- 1.1 Définitions et concepts de base 1
- 1.2 Éléments de la problématique 2
- 1.3 Objectifs de recherche 4
- 1.4 Plan du mémoire 4

CHAPITRE 2 REVUE DE LITTÉRATURE 6
- 2.1 Stratégies de modélisation de l'environnement 8
- 2.2 Histoire de la « cartographie robotique » 11
- 2.3 Tour des méthodes classiques 13
  - 2.3.1 Filtres de Kalman 13
  - 2.3.2 Grilles d'occupation 14
  - 2.3.3 Filtres Rao-Blackwellized 15
- 2.4 Aspect probabiliste 18
- 2.5 Les difficultés majeures 20
  - 2.5.1 Problème de corrélation 20
  - 2.5.2 Complexité en terme de dimension 21
  - 2.5.3 Complexité de la modélisation 22

2.5.4 Problème de correspondance . . . 22
2.5.5 Chemin d'exploration . . . 24
2.5.6 Changements de l'environnement . . . 25
2.5.7 Obstacles transparents . . . 26
2.6 Conclusion et pistes récentes . . . 26

CHAPITRE 3 ENRICHISSEMENT INTELLIGENT PAR DONNÉES SONAR DES CARTES GÉNÉRÉES PAR DÉTECTIONS LASER VIA SLAM . . . 29
3.1 Limites de la cartographie disponible . . . 29
3.1.1 Architecture du système à optimiser . . . 30
3.1.2 Algorithme *GMapping* . . . 33
3.2 Architecture de solution . . . 36
3.3 Algorithme solution . . . 37
3.3.1 Entrées et Sortie . . . 37
3.3.2 Étapes de l'algorithme . . . 37
3.4 Simulation et Choix de la méthode de construction de carte . . . 39
3.4.1 Les troix méthodes . . . 40
3.4.2 Récolte et préparation des données expérimentales . . . 41
3.4.3 Premiers résultats de simulation . . . 43
3.4.4 Comparaison expérimentale des méthodes . . . 45
3.5 Éléments importants de l'implémentation . . . 48
3.5.1 Lecture et fréquences de publications . . . 48
3.5.2 Formats des cartes . . . 49
3.6 Résultats de test en système réel . . . 49
3.6.1 Test en laboratoire . . . 49
3.6.2 Test en milieu public . . . 50
3.6.3 Discussion . . . 51

CHAPITRE 4 CONSTRUCTION AUTOMATIQUE DE CARTES EN GRILLES D'OCCUPATION A PARTIR DE PLANS D'ARCHITECTURE . . . 54
4.1 Plans d'architecture et formats . . . 54
4.1.1 Notions générales . . . 54
4.1.2 Structure du format *DXF* . . . 55
4.2 Description de la solution . . . 57
4.2.1 Entrées et Sorties . . . 57
4.2.2 Structure de l'algorithme . . . 58
4.2.3 Pré-traitement du plan . . . 59

4.2.4 Construction de la carte 61
4.3 Programme et Application 64
4.3.1 Éléments importants de l'implémentation 64
4.3.2 Exemples de cartes d'étages construites 65
4.4 Utilisation de la carte construite en navigation 71
4.4.1 Navigation manuelle et auto-localisation 71
4.4.2 Navigation point à point dans la carte 79
4.4.3 Enrichissement intelligent par les détections des télémètres ultrason . . 82
4.5 Conclusion 85

CHAPITRE 5 CONCLUSION 86
5.1 Synthèse des travaux 86
5.2 Limitations des solutions proposées 87
5.2.1 Limitations de la solution d'enrichissement de cartes 87
5.2.2 Limitations de la solution de génération automatique de cartes 88
5.3 Améliorations futures 88

RÉFÉRENCES 89

ANNEXES 93
A.1 Première méthode : « *Probabilistic approach* » 93
A.2 Deuxième méthode : « *Evidence theoretic approach* » 94
A.3 Troixième méthode : « *Possibilistic approach* » 95

LISTE DES TABLEAUX

Tableau 2.1 Comparaison des stratégies de cartographie 11
Tableau 4.1 Éléments graphiques, leurs noms d'objet et leurs codes utiles 62

LISTE DES FIGURES

Figure 2.1 Exemple simple d'expérience fictive de localisation en exploitant un modèle de l'environnement 9
Figure 2.2 Planification de chemin dans une carte (carte : noir et blanc, obstacle dynamique : rouge, plan global : bleu, plan local : vert) 10
Figure 2.3 Simulation d'une construction avec poses connues d'une carte en grille d'occupation 15
Figure 2.4 5000 points aléatoirement dessinés selon une loi uniforme sur la carte du monde ; les points sur la terre sont blancs, ceux sur les océans sont noirs. 16
Figure 2.5 Nuage de point de la simulation du mouvement élémentaire d'un robot pour 1000 échantillons avec différents paramètres de bruit (rectangle bleu : pose de départ, rectangle rouge : pose d'arrivée, cercle noir : centre de l'arc du mouvement élémentaire) 23
Figure 2.6 Cartographie de l'étage 1 du pavillon Lassonde de l'École Polytechnique avec mauvaise gestion du problème de correspondance : Carte prise de El-Fathi (2012) sous permission de l'auteur 24
Figure 2.7 Cartographie d'un milieu contrôlé en présence d'obstacles transparents (représentés en rouge) en utilisant les télémètres laser seulement : Carte prise de El-Fathi (2012) sous permission de l'auteur 27
Figure 2.8 Cartographie d'un milieu contrôlé en présence d'obstacles transparents (représentés en rouge) en utilisant les télémètres laser et ultrason (carte à droite) : Cartes prise de El-Fathi (2012) sous permission de l'auteur . 27
Figure 3.1 Architecture de la cartographie à optimiser 32
Figure 3.2 Architecture proposée de la cartographie améliorée 33
Figure 3.3 Les composants majeurs du *GMapping* 34
Figure 3.4 Architecture de solution 36
Figure 3.5 Figure explicative de la détermination de la zone de traitement où $1.5m$ représente la portée maximale de nos sonars 38
Figure 3.6 Figure montrant les coordonnées polaires d'une cellule dans le repère du sonar 41
Figure 3.7 Modèle inverse de la première méthode pour une mesure de $0.5m$ . . . 42
Figure 3.8 Modèle inverse de la deuxième et troixième méthodes pour une mesure de $0.5m$ 43

Figure 3.9 Premiers résultats de simulation du module d'enrichissement : carte-laser avec obstacles transparents rajoutés dessus en bleu (en haut à gauche) et les cartes enrichies pour les 3 méthodes 44
Figure 3.10 Cartes-laser obtenues dans un milieu contrôlé sans et avec obstacles transparents 45
Figure 3.11 Résultats de simulation pour les troix méthodes de calcul probabiliste . 46
Figure 3.12 Scores des différentes méthodes de calcul probabiliste en simulation avec des données récoltées sur le FRMI en déplacement dans un milieu contrôlé 47
Figure 3.13 Résultat de test en labo : carte-laser 50
Figure 3.14 Carte-laser où les obstacles transparents sont indiqués 50
Figure 3.15 Résultat de test en labo : carte-sonar 51
Figure 3.16 Résultat de test à l'étage 5 du pavillon Lassonde : carte-laser 52
Figure 3.17 Résultat de test à l'étage 5 du pavillon Lassonde : carte-sonar 52
Figure 4.1 Structure générale d'un fichier DXF : Mise en valeur des éléments importants 56
Figure 4.2 Structure du module et flux de données 58
Figure 4.3 Structure du composant de pré-traitement 61
Figure 4.4 Zone de dessin d'une ligne 64
Figure 4.5 Zone de dessin d'un cercle 64
Figure 4.6 Visualisation du plan d'architecture original du cinquème étage du pavillon Lassonde de Polytechnique 66
Figure 4.7 Liste complète des calques du plan de l'étage 5 et liste des calques choisis 67
Figure 4.8 Plan simplifié de l'étage 5 68
Figure 4.9 Carte de navigation de l'étage 5 générée par notre programme avec une résolution de 0.05m 69
Figure 4.10 Visualisation du plan d'architecture original du premier étage du pavillon Lassonde de Polytechnique 72
Figure 4.11 Carte de navigation du premier étage générée par notre programme avec une résolution de 0.05m 73
Figure 4.12 Visualisation du plan d'architecture original du troixième étage du pavillon Lassonde de Polytechnique 74
Figure 4.13 Carte de navigation du troixème étage générée par notre programme avec une résolution de 0.05m 75
Figure 4.14 Secteurs de test de la navigation manuelle dans la carte générée du cinquième étage 76

Figure 4.15 Navigation manuelle avec auto-localisation dans une carte générée automatiquement : position initiale sans décalage 77
Figure 4.16 Navigation manuelle avec auto-localisation dans une carte générée automatiquement : position initiale avec décalage 78
Figure 4.17 Schéma des tests de la navigation point à point dans la carte construite du cinquième étage du pavillon Lassonde 80
Figure 4.18 Captures des tests de la navigation point à point dans la carte construite du cinquième étage du pavillon Lassonde 81
Figure 4.19 Carte construite automatiquement du cinquième étage où deux obstacles manquants sont indiqués manuellement 82
Figure 4.20 Carptures montrant le FRMI en train d'enrichir la carte de l'étage 5 : dessin d'un obstacle transparent et d'un obstacle opaque 83
Figure 4.21 Carte construite automatiquement du cinquième étage, enrichie par le module d'enrichissement de carte par données sonar, pour compléter les deux obstacles manquants 84

LISTE DES ANNEXES

Annexe A Trois méthodes de calcul probabiliste de grilles d'occupation par les mesures des sonars 93

LISTE DES SIGLES ET ABRÉVIATIONS

AMCL	Adaptive Monte Carlo Localization
CAO	Conception Assistée par Ordinateur
DXF	Drawing eXchange Format
FRMI	Fauteuil Roulant Motorisé Intelligent
ROS	Robotic Operating System
SLAM	Simultaneous Localization And Mapping

CHAPITRE 1

INTRODUCTION

Quiconque a pris le métro a certes remarqué ces cartes affichées à l'entrée ou à la sortie du métro et qui indiquent souvent leurs emplacements dans l'environnement par la mention « *Vous êtes ici* ». Des cartes utiles de ce genre sont également trouvées à des points particuliers de la plupart des milieus publiques tels que les centres commerciaux, les hôpitaux, les hôtels, etc. Ces cartes font depuis longtemps partie de notre quotidien au point que nous les utilisons de nos jours de façon instinctive sans méditer sur leur énorme utilité pour nos déplacements et encore moins sur leur rôle dans le mécanisme de « navigation » de l'Homme.

Même en absence de cartes affichées, il existe une carte qui est toujours présente pour nous « guider » : notre carte mentale ou *Carte Cognitive*, concept introduit par *Edward Tolman* en 1948 (Tolman (1948)). On pense que l'hippocampe du cerveau humain est la partie responsable de la cartographie cognitive des endroits visités.

Le besoin d'une carte est alors claire pour la « navigation » de l'homme. Ce besoin est encore plus criant dans le cas des utilisateurs de fauteuils roulants où ces fauteuils se chargent totalement ou partiellement de la navigation globale (manuelle ou automatique) de la personne de mobilité réduite. Il serait ainsi intéressant de contribuer au domaine de recherche visant à apporter plus de facilité et de confort à l'expérience de navigation de cette catégorie de patients.

1.1 Définitions et concepts de base

Un projet de réalisation d'un prototype de fauteuil roulant motorisé intelligent (FRMI) avait réuni depuis quelques années les efforts de chercheurs et des étudiants aux cycles supérieurs de trois universités du Québec : École Polytechnique de Montréal, Université de McGill et Université de Montréal.

Le FRMI est un fauteuil roulant motorisé (FRM) équipé de capteurs (encodeurs pour mesures d'odométrie, 2 télémètres Laser, 6 télémètres ultrason et une caméra Kinect) et d'un système de navigation qui se base sur un ensemble de modules communiquant entre eux El-Fathi (2012). Ces modules permettent l'automatisation de plusieurs tâches :

- Locales : suivi de mur, évitement d'obstacles... On parle donc de *navigation locale*.
- Globales : navigation point à point dans une carte, autolocalisation dans une carte, localisation et cartographie simultanées... On parle donc de *navigation globale*.

Il est ainsi un cas particulier de robot mobile. D'où l'importance de la contribution de la recherche dans ce projet au domaine scientifique de la robotique mobile.

Les modules du FRMI sont implémentés dans un environnement *ROS* (*Robot Operating System*). ROS est un méta système d'exploitation conçu pour faciliter la réalisation d'applications robotiques complexes. c'est un système puissant qui a plusieurs avantages comme la disponibilité d'une large bibliothèque de pilotes des différents capteurs et actionneurs, l'utilisation pratique d'outils très avancés tels que l'outil **tf** permettant d'automatiser le calcul des relations et des transformations entre les repères, la gestion efficace et synchrone de la communication entre les modules (appelés nœuds dans ROS), etc.

La navigation globale du FRMI est complétement dépendante de la modélisation de l'environnement. À cet effet, une carte est généralement construite par utilisation d'un module classique de SLAM.

Le *SLAM* désigne le problème de localisation et cartographie simultanées. Il est également devenu par abus de language le nom de la méthode utilisée par les robots et les véhicules mobiles qui vise à répondre au problème de SLAM, c'est à dire, à construire une carte dans un environnement inconnu, ou pour mettre à jour une carte d'un environnement connu, tout en gardant une trace de leur position actuelle.

1.2 Éléments de la problématique

Comme le FRMI est conçu pour faciliter les déplacements des personnes à mobilité réduite, son environnement de navigation en tant que robot devrait être étudié et modélisé en se référant aux milieux généralement visités par un utilisateur de fauteuil roulant. L'environnement en question est du coup caractérisé par un ensemble de particularités qui contrôlent les possibilités et dirigent les idées et les perspectives d'optimisation de la navigation globale de ce robot particulier.

Les particularités les plus importantes de l'environnement du FRMI sont :

- les milieux visités sont généralement intérieurs dominés par les structures construites par l'Homme.
- les obstacles et les éléments qui sont rencontrés sont souvent semblables et répétitifs dans tout les milieux (portes, couloirs, piliers, murs, ...).
- les bâtiments modernes connaissent l'utilisation plus fréquente des vitres et des polymères transparents.
- à la différence des autres environnements beaucoup plus « naturels » et beaucoup moins contrôlé par l'Homme, l'environnement du FRMI a souvent plusieurs modélisations et cartes réalisées pour diverses raisons. En particulier, on est sensé trouver, presque

toujours, un plan architectural de l'étage où se passe la navigation.
- la plupart des milieux sont publiques et donc souvent encombrés par les passants (nombreux obstacles dynamiques).

On peut ainsi dégager deux points sur lesquels est axée la recherche qui vise à optimiser la navigation globale :

- Si nous savons que l'algorithme utilisé par le FRMI pour répondre au problème du SLAM profite des détections des télémètres laser uniquement pour représenter les obstacles, les obstacles transparents sont ainsi non représentés car non détectés. Comment pouvons nous alors remédier à ce problème vu le danger que représente un obstacle non pris en compte dans la carte ?
- Est-il possible d'éviter l'exploration de l'environnement pour construire sa carte ? On pense que, logiquement, une construction d'une carte de l'environnement qui s'affranchit de l'exploration doit forcément passer par l'exploitation d'un autre modèle disponible de cet environnement. Les plans d'architecture semblent être les meilleurs candidats. Y a-t-il alors un moyen de les exploiter pour construire une carte utile pour la navigation du FRMI ?

C'est autour de ces deux questions essentielles que nous allons réfléchir à des solutions qui permettent de rendre la navigation plus performante. Regardons alors de plus près ces deux axes pour avancer des éléments de réponse.

Premier axe : Pour détecter les vitres, il faut utiliser des capteurs adaptés aux obstacles transparents. Les mesures des sonars fournissent alors les informations nécessaires à la représentation de ces obstacles. Comme le FRMI dispose bien d'un module efficace de localisation et cartographie simultanées, le problème des obstacles transparents peut être ramené à l'exploitation de la carte construite par les lasers en plus de la correction continue de la pose du FRMI pour trouver le moyen d'enrichir cette carte avec les mesures des sonars au bon moment et au bon emplacement.

Le module d'autolocalisation nous pousse aussi à élargir l'usage d'une éventuelle solution d'enrichissement au cas d'une carte statique en exploitant dans ce cas la correction continue de la pose uniquement.

Deuxième axe : La deuxième question s'intéresse au problème spécifique de création d'une carte à partir d'un plan architectural. Ici d'autres questions sont à poser pour clarifier la démarche à suivre dans la recherche d'une technique qui résoud le problème :

- Les plans se présentent sous plusieurs formats différents. Quel format choisir ?
- Quels éléments faut-il extraire des plans et suivant quelle stratégie ? Quel est le rôle

que joue le choix du format dans la stratégie de construction (utilisation des méthodes de traitement d'image ou exploitation des avantages des formats vectoriels)
- Quelle précision de construction et quelle résolution choisir pour assurer la compatibilité de la carte avec les autres modules de navigation ?
- Quels tests faut-il prévoir et organiser pour valider l'utilité de la carte construite pour la navigation globale du FRMI ?

1.3 Objectifs de recherche

Développer deux techniques d'optimisation de la navigation globale basée sur une carte est l'objectif ultime de notre recherche. La réalisation de cet objectif passe par deux autres sous-objectifs plus spécifiques :

- concevoir et implémenter un composant compatible avec les modules de SLAM et d'autolocalisation classiques, en particulier le *GMapping* et le module d'auto-localisation implémentés sur ROS et utilisés dans le système de navigation du FRMI, et qui permet d'enrichir les cartes construites avec les mesures des lasers uniquement par les mesures des sonars afin de compléter les obstacles transparents non détectés. Pour atteindre ce but, nous allons : étudier l'algorithme de SLAM classique pour justifier l'architecture du composant à créer, simuler et tester les méthodes d'enrichissement pour faire un choix de la méthode à implémenter, et enfin tester le composant en vue de sa validation.
- proposer et implémenter une méthode de construction d'une carte supportée par le système de navigation du FRMI à partir d'un plan architectural d'étage. Ceci passe par : le choix d'un format de plans et l'établissement de la méthode d'extraction d'éléments adaptée, l'implémentation de la méthode, la réalisation des tests et la validation par application de tâches de navigation.

1.4 Plan du mémoire

Dans ce mémoire, nous présentons d'abord une revue de littérature qui se veut riche et complète de la cartographie robotique. En effet, nous exposons les particularités principales qui font l'intérêt de la recherche qui vise à modéliser l'environnement d'un robot pour, ensuite, énumérer et comparer les stratégies de modélisation. Puis, en nous basant sur l'histoire de la cartographie robotique, nous proposons des explications concises des difficultés les plus importantes qu'ont historiquement rencontrées les chercheurs et des méthodes que nous avons jugées incontournables tout en mettant au passage le point sur l'aspect probabiliste commun aux solutions classiques de cartographie.

Dans un second lieu, nous proposons une première technique d'optimisation de la navigation globale du FRMI : l'enrichissement par les mesures des sonars des cartes générées par détections laser en utilisant un module classique de SLAM. Cette technique est exposée en détails de la conception par étude du module de SLAM jusqu'à l'implémentation et les tests en milieu réel en passant par la construction de l'algorithme et les tests en simulation.

Un autre chapitre est consacré à une deuxième technique d'optimisation de la navigation globale : la construction automatique de grilles d'occupation à partir de plans architecturaux. Nous montrons pour cette technique, en plus des notions de bases exploitées et de la structure de l'algorithme solution, un ensemble d'applications poussées, testées sur le FRMI, du système de navigation qui exploite une carte d'étage construite par cette technique.

Enfin, nous présentons en conclusion la synthèse des travaux ainsi que les limites des techniques proposées et nous proposons des améliorations possibles.

CHAPITRE 2

REVUE DE LITTÉRATURE

Dans le domaine de la robotique, la cartographie robotique est le problème qui s'intéresse à s'approprier un modèle spatial de l'environnement du robot qui soit supporté par son système et utile pour sa navigation.

Il y a de bonnes raisons qui laissent penser que la cartographie est parmi les sujets de recherche d'actualité en robotique auquels les chercheurs s'intéressent le plus.

Pourquoi la cartographie robotique suscite-t-elle autant d'intérêt ?

La réponse à cette question passe par la connaissance de 2 aspects importants du sujet :

- Les défis et les difficultés majeures qui se rapportent à la cartographie face au niveau d'avancement des outils théoriques à disposition.
- Pourquoi disposer d'une carte est-il si important ?

Le premier aspect justifie le développement historique de l'intérêt croissant au problème de création de cartes des environnements des robots.

En effet, la recherche a connu un échange mutuel entre l'établissement d'algorithmes pratiques de cartographie et le développement du paradigme probabiliste de traitement des problèmes complexes. Ainsi, les difficultés de la cartographie (2.5) ont justifiés le recours aux approches probabilistes (2.4) et la domination quasi-totale de celles-ci dans le domaine de la robotique plus généralement. Et en contre partie, les méthodes probabilistes (2.3) ont trouvé une occasion d'application concrète et très intéressante.

Par exemple, le rapport technique de Doucet (1998) faisant le tour « des méthodes séquentielles basées sur la simulation pour la réalisation de filtres pour les modèles dynamiques non linéaires et non gaussiens » décrit le filtre *Rao-Blackwellized*[1]. Ce filtre est proposé comme une solution au problème de cartographie et localisation simultanées (connu sous le nom de *SLAM*[2]).

Les articles Murphy (1999) et Doucet *et al.* (2000) sont les premiers à introduire des systèmes efficaces qui l'utilisent pour résoudre ce problème. Ensuite, d'autres travaux ont focalisé sur l'amélioration de ce filtre dans le cadre du SLAM. Grisetti *et al.* (2006) ont ainsi travaillé sur l'optimisation du filtre *Rao-Blackwellized*, toujours dans le cadre de l'application au problème de *SLAM*, pour diminuer le nombre de ses particules et réduire le risque de leur appauvris-

1. Nous en parlerons avec plus de détails en 2.3.2
2. Simultaneous Localization And Mapping

sement [3].

Le second aspect du sujet traduit la place importante du modèle de l'environnement pour la robotique.

En effet, le besoin d'une carte en robotique peut, selon nos lectures sur le sujet, se résumer en 3 motifs principaux :

- Le besoin de visibilité et d'information : La carte fournit une expérience de navigation plus visuelle et donc plus intuitive (aspect cognitif). Elle constitue également une source d'informations utiles, que ce soit pour le robot ou pour son utilisateur, sur la topologie du milieu, la disposition de ses obstacles, l'emplacement des objets utiles etc...
- Le besoin de localisation : L'Homme utilise des cartes pour se localiser par rapport à son environnement et donc se déplacer vers sa destination à partir de sa position actuelle. Ce besoin est le même pour le robot. La localisation est alors généralement effectuée par comparaison du modèle du milieu à des caractéristiques particulières, locales (obstacles ...) ou globales (force d'un signal, niveau par rapport au sol...), détectées par les capteurs du robot.

  La figure 2.1 illustre ce besoin de localisation par un exemple simple imaginé où la position du robot, capable de reconnaître un pot de fleurs dans son environnement, est estimée selon une logique probabiliste par comparaison à une carte mémorisée.

  À sa position initiale, toutes les positions sont équiprobables. Ensuite, dès qu'il détecte un pot de fleurs, toutes les positions des pots dans sa carte sont les plus probables. Et finalement, connaissant la distance parcourue avant de rencontrer le second pot, le robot détermine sa position sans ambiguïté.
- Le besoin de navigation globale : La planification de chemin, pour effectuer une navigation point à point automatique par exemple, est réalisée généralement en deux temps : d'abord, une planification d'un chemin global relie la pose [4] initiale à la destination finale du robot.

  Puis, une planification locale continue en temps réel maintient le robot sur le plan global tout en évitant les obstacles dynamiques rencontrés.

  Comme les capteurs qui détectent les obstacles sont de portées limitées, d'autant plus que les obstacles les plus proches cachent ceux qui sont plus loins, la planification globale n'est efficace, et dans certains cas n'est utile, que lorsqu'elle tient compte des obstacles statiques qui confinent le meilleur chemin global. D'où, la nécessité d'un modèle préalable de l'environnement contenant ces obstacles statiques, ou encore une carte

3. En anglais, « particle depletion ». Ici des notions que nous définirons plus loin dans ce chapitre sont évoquées juste pour enrichir l'exemple.

4. Dans ce rapport, nous appelons pose l'ensemble d'un point et d'une orientation permettant de connaitre complètement la dispostion d'un objet rigide non ponctuel dans un plan

du milieu.

La figure 2.2 illustre cette planification de chemin dans une carte de l'environnement à l'aide d'un schéma d'exemple simple.

Ainsi, la matière de la recherche de la cartographie robotique est riche et importante. Elle traduit des années d'accumulation et de développement des différentes techniques et méthodes. De ce fait, certains chercheurs se sont intéressés à ce sujet plus que les autres sujets et ils ont réalisés alors des documents très instructifs sur la cartographie.
Parmi ces productions, on cite en particulier le *survey* de Thrun (2002) : la revue de littérature la plus complète que nous avons trouvée, et la partie III intitulée *Mapping* du livre "Probabilistic Robotics" (Thrun *et al.* (2005)) exposant les méthodes et les algorithmes classiques de cartographie robotique.

2.1 Stratégies de modélisation de l'environnement

Pour modéliser l'environnement du robot, 3 stratégies sont utilisées.

D'abord, le moyen le plus direct d'avoir une carte d'un milieu donné est de la construire « manuellement » en s'inspirant du milieu directement.
« Construire manuellement » signifie que l'agent humain (opérateur ou utilisateur du robot...) se charge de tout le travail de construction : les mesures des dimensions, distances et orientations de tous les obstacles, leur modélisation, les calculs liés à la résolution ou l'échelle adoptée, les différents traitements d'agencement et de construction effective (éventuellement par l'intermédiaire d'un outil de conception assistée par ordinateur) des éléments, etc. Notons que ce travail énorme et pénible de construire entièrement une carte est de difficulté « exponentiellement » croissante avec la complexité du milieu à cartographier. D'autant plus qu'en absence de support de plan ou de carte déjà réalisée [5], le travail est encore plus difficile car nécessitant l'exploration du milieu et une certaine représentation par l'esprit du constructeur « humain ».
Un autre inconvénient de cette stratégie est l'absence totale (vu qu'elle est « manuelle ») de toute automatisation de la construction. Chaque environnement est une nouvelle expérience où le constructeur est obligé de passer par toutes les étapes à chaque fois. Bien évidemment, certaines étapes peuvent être automatisées pour diminuer la charge de travail du constructeur. Mais, la prise des données nécessaires sur les obstacles physiques du milieu implique pour assurer son automatisation :

– soit un mécanisme intelligent d'interprétation des obstacles représentés dans une autre carte à un format non supporté par le robot. Là, nous touchons la frontière avec la

5. on parle d'une carte disponible sous un format non-supporté par le robot car sinon le plus facile est d'utiliser la carte disponible directement

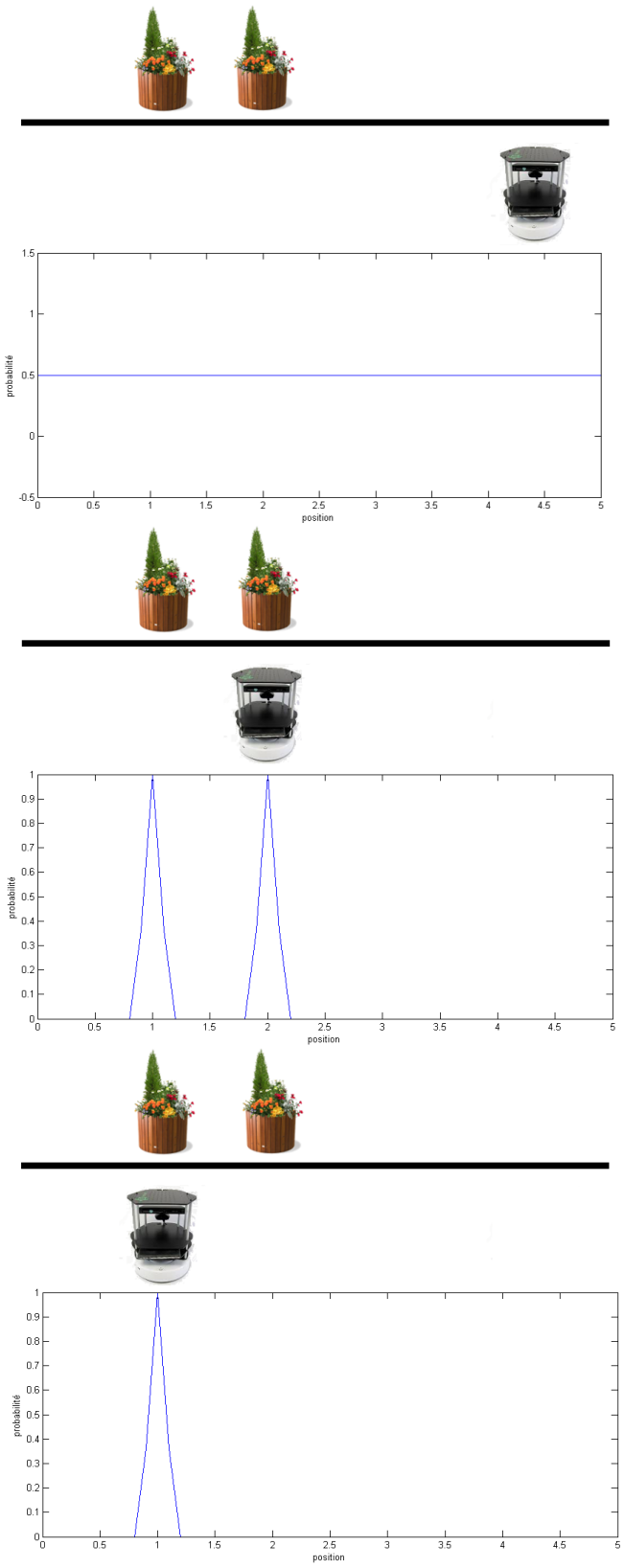

Figure 2.1 Exemple simple d'expérience fictive de localisation en exploitant un modèle de l'environnement

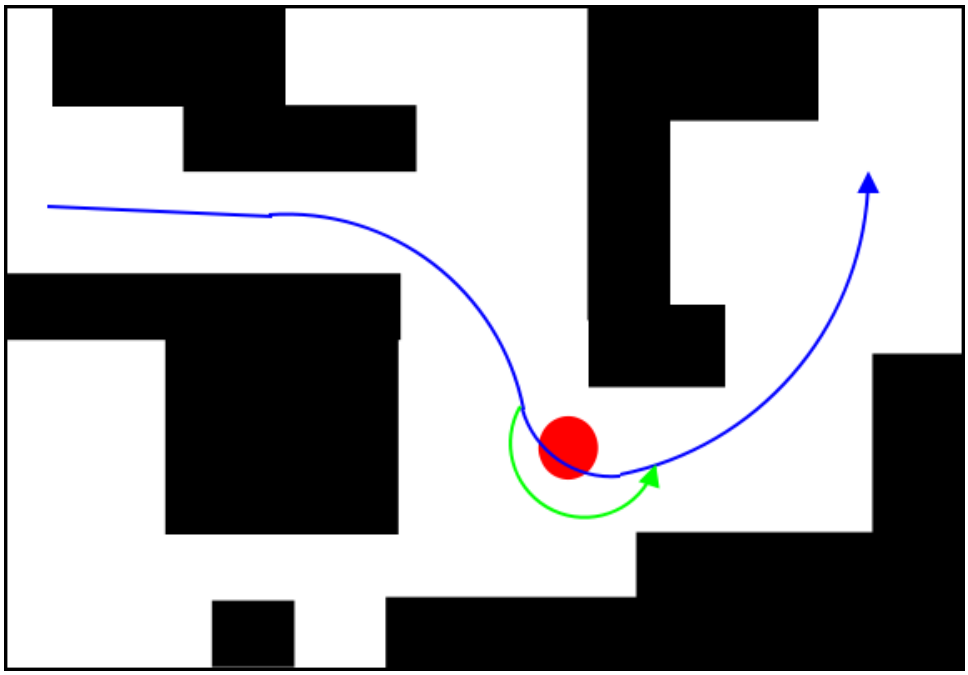

Figure 2.2 Planification de chemin dans une carte (carte : noir et blanc, obstacle dynamique : rouge, plan global : bleu, plan local : vert)

troisième stratégie de modélisation du milieu que nous allons présenter plus bas.

– soit une « exploration » du milieu au moyen d'un robot muni des capteurs nécessaires pour le collecte de ces données. Là, il s'agit de la frontière avec la deuxième stratégie que nous allons introduire plus bas.

L'avantage fondamental de cette stratégie est la gestion quasi-parfaite (car directe par l'esprit de l'Homme) de la fermeture des cycles (problème de correspondance 2.5.4). Le constructeur reconnaît généralement sans difficultés un endroit qu'il a déjà vu.

La deuxième stratégie consiste en la construction automatique de la carte à partir des données collectées par exploration du milieu. L'exemple classique de construction sous cette catégorie de méthodes est celui du SLAM où les données prises en temps réel, tout en explorant l'environnement, sont exploitées dans l'élargissement et l'enrichissement, en temps réel aussi, de la carte.

La plupart des travaux réalisés jusqu'aujourd'hui rentrent dans cette catégorie : Dissanayake *et al.* (2000), Durrant-Whyte *et al.* (2001), Leonard et Feder (1999), Thrun *et al.* (1998).

La troisième stratégie de cartographie est la moins présente dans la littérature. La carte n'est pas construite par utilisation directe du robot. Le modèle est plutôt construit par lecture et interprétation automatique d'une autre carte disponible (un plan architectural dans tous les cas que nous avons rencontrés).

Schafer *et al.* (2011) et Farhan *et al.* (2011) sont les deux seuls exemples de travaux, que nous avons trouvés, qui proposent des solutions de cette catégorie.

Ces deux dernières stratégies de cartographie automatique présentent chacune des points forts et des faiblesses. Le tableau 2.1 aide à mieux les comparer entre elles d'un côté et par rapport à la stratégie manuelle d'un autre côté.

Tableau 2.1 Comparaison des stratégies de cartographie

Stratégie	Avantages	Faiblesses
manuelle	» gère bien le problème de correspondance	» grande charge de travail » effort de représentation mentale » conseillée pour un nombre limité de cartes
par exploration et collecte de données	» cartes riches et complètes » précision des dimensions	» temps de construction élevé » nécessité d'exploration de tout le milieu » consommation d'énérgie et de CPU » problème de corrélation (en cas de SLAM) » dépendance des capteurs et de la nature des obstacles » problème de correspondance
par interprétation de plans	» efficace pour le problème de correspondance » rapide	» dépend de la qualité du plan » cartes généralement incomplètes (poubelles, armoires, etc. non représentées)

2.2 Histoire de la « cartographie robotique »

Durant les années 80, des travaux de recherche ont abouti à l'introduction des grilles d'occupation. Cette méthode consiste à modéliser l'environnement par une grille de cellules où on accorde à chaque cellule une grandeur qui quantifie sa probabilité d'occupation. Cette probabilité est mise à jour de façon incrémentale grâce aux observations des capteurs. Elfes (Elfes (1987, 1989)) et Moravec (Moravec (1988); Moravec et Elfes (1984)) étaient les premiers à introduire cette méthode de construction de cartes métriques. Le problème résolu par leurs algorithmes est celui de la cartographie avec poses connues.

Les capteurs utilisés pour détecter les obstacles de l'environnement étaient les sonars à l'époque. Du coup, ces premiers travaux et d'autres qui ont permis ultérieurement de comparer les méthodes de calcul probabiliste des grilles d'occupation avec les mesures des sonars

(Gambino *et al.* (1996); Ribo et Pinz (2001)) constituent les références de la cartographie sonar et des modèles de perception de ce type de capteurs.

Parallèlement à l'apparition des grilles d'occupation, les premières solutions au problème de SLAM en robotique ont vu le jour : les approches basées sur le filtre de Kalman étendu Cheeseman et Smith (1986); Smith *et al.* (1990); Moutarlier et Chatila (1989a,b). Dans ces approches, l'incertitude est supposée gaussienne et additionnelle, les modèles de perception et de mouvement sont linéarisés par développement en série de Taylor et l'état à estimer est constitué de la pose du robot en plus de la carte qui est en fait un ensemble de positions d'objets caractéristiques. Beaucoup de travaux ont ultérieurement permis de développer l'approche par filtre de Kalman (Castellanos *et al.* (1999); Leonard et Feder (1999); Newman (2000); Dissanayake *et al.* (2001); Durrant-Whyte *et al.* (2001)).

D'autres approches de cartographie robotique utilisent :

- l'algorithme d'expectation-maximisation (EM) de Dempster *et al.* (1977) (Thrun *et al.* (1998); Thrun (2001)) : un algorithme qui itère *Expectation* où les poses les plus probables du robot sont estimées dans une carte donnée, et, *Maximisation* où la carte la plus probable est calculée étant donné un ensemble de poses représentant le chemin le plus probable du robot.
- l'optimisation non-linéaire (Olson *et al.* (2006, 2007); Grisetti *et al.* (2007)) où une méthode d'optimisation itérative (moindres carrés, méthode du gradient, ...) est utilisée sans aucune linéarisation pour trouver le maximum global de la probabilité d'obtention des données mesurées à chaque itération sur tout l'ensemble (discrétisé) des poses possibles.
- les filtres à particules Rao-Blackwellized (Murphy (1999); Doucet *et al.* (2000); Montemerlo (2003); Stachniss *et al.* (2005); Grisetti *et al.* (2005, 2006)) où le filtre à particules, sous une hypothèse justifiée (la *Rao-Blackwellization*), nous évite de calculer toute la distribution de probabilité représentant le problème de SLAM en l'approximant par un ensemble d'échantillons générés aléatoirement.
- la méthode iSAM (*Incremental Smoothing and Mapping*) (Kaess *et al.* (2007, 2008)) où, sans aucune approximation, la trajectoire du robot ainsi que la carte de l'environnement sont obtenues en temps réel par lissage (*smoothing*). La matrice de lissage qui est naturellement creuse est « QR-décomposée » (Décomposition QR), pour avoir accès aux informations liées à la gestion du problème de correspondance, et est mise à jour sous cette forme factorisée à chaque nouvelle mesure.

Ces approches sont venues élargir la gamme d'approches probabilistes. La probabilité domine clairement les méthodes des chercheurs en cartographie robotique. Le problème de SLAM est resté au centre des intérêts de recherche pour la plupart des travaux évoqués.

Des productions récentes (Schafer *et al.* (2011); Farhan *et al.* (2011)) exposent des méthodes de construction de cartes à partir de plan architecturaux. Schafer *et al.* proposent un algorithme permettant d'extraire les portes, les pièces et les escaliers dans un plan au format DXF pour construire une carte topologique de l'environnement. Quant à Farhan *et al.*, ils traitent le plan au format d'une image grâce à des techniques de traitement d'images pour aboutir à la carte.

2.3 Tour des méthodes classiques

2.3.1 Filtres de Kalman

Les approches basées sur des filtres de Kalman sont classiques en cartographie robotique selon Thrun. La multitude des travaux qui s'intéressent à ces approches confirme cette affirmation : Durrant-Whyte *et al.* (2001), Castellanos *et al.* (1999), Dissanayake *et al.* (2001), Leonard et Feder (1999), Newman (2000), etc.

La méthode repose sur les hypothèses suivantes :

- les environnements sont modélisés par des cartes d'objets caractéristiques (des sortes de jalons ou points correspondant à des formes reconnues par le robot) [6]
- les modèles (perception/mouvement) sont linéaires avec bruit gaussien ajouté.
- l'incertitude initiale est gaussienne.

En notant l'état à estimer $e_t = (x_t, m)^T$ où x_t désigne la pose du robot à l'instant t et m la carte de l'environnement, le filtre est ainsi implémenté en utilisant les équations standards des filtres de Kalman (Kalman (1960)). Les modèles de perception et de mouvement sont bien évidemment non linéaires en réalité mais une linéarisation par développement en série de Taylor permet de proposer une approximation qui remplit l'hypothèse.
Ces modèles s'écrivent, en désignant par u la commande de mouvement (ou l'odométrie) et par z la mesure de perception :

$$\begin{aligned} p(e|u, e') &= Ae' + Bu + \epsilon_u \\ p(z|e) &= Ce + \epsilon_z \end{aligned}$$

où ϵ_u et ϵ_z sont des bruits blancs gaussiens.
Du coup, il s'agit en réalité d'un filtre de Kalman étendu (EKF).

6. « feature-based maps » en anglais

2.3.2 Grilles d'occupation

Introduit pour la première fois par Elfes et Moravec (Elfes (1989), Moravec (1988)), l'algorithme des cartes en grilles d'occupation [7] est devenu un classique de la cartographie robotique. Il donne une méthode pratique de résolution du problème de cartographie avec poses connues qui est un problème plus simple que le SLAM.
Le principe est de construire une carte métrique de l'environnement en décomposant la surface explorée en cellules. Une probabilité d'occupation $p(m_i|z_{1:t}, x_{1:t})$ est associée à chaque cellule (on adopte les mêmes notations que 2.3.1). Et on fait l'hypothèse que les probabilités d'occupation des cellules sont indépendantes les unes des autres. Le théorème de Bayes permet d'écrire :

$$\frac{p(m_i|z_{1:t}, x_{1:t})}{1 - p(m_i|z_{1:t}, x_{1:t})} = \frac{p(m_i|z_{1:t-1}, x_{1:t-1})}{1 - p(m_i|z_{1:t-1}, x_{1:t-1})} \cdot \frac{p(m_i|z_t, x_t)}{1 - p(m_i|z_t, x_t)} \cdot \frac{1 - p(m_i)}{p(m_i)}$$

où $p(m_i)$ est la probabilité préalable à l'occupation.
Si on pose :

$$L_{t,i} = \log \frac{p(m_i|z_{1:t}, x_{1:t})}{1 - p(m_i|z_{1:t}, x_{1:t})}$$

On obtient alors la loi d'évolution itérative suivante :

$$L_{t,i} = L_{t-1,i} + \log \frac{p(m_i|z_t, x_t)}{1 - p(m_i|z_t, x_t)} - \log \frac{p(m_i)}{1 - p(m_i)}$$

Il existe des algorithmes qui permettent de proposer un calcul de la probabilité $p(m_i|z_t, x_t)$ dépendemment du capteur utilisé. Nous avons ici, en guise d'illustration de cette approche, choisi l'algorithme 9.2 proposé par Thrun *et al.* (2005) pour le calcul de cette probabilité. Il s'agit d'un modèle inverse de perception conçu pour les télémètres laser. Nous simulons donc la cartographie d'une pièce circulaire explorée par un robot placé à son centre et qui tourne autour de lui-même dans le sens des aiguilles d'une montre en prenant à chaque 9 degrés des mesures. Les mesures sont générés par lancement de rayons vers l'avant de façon symétrique par rapport au robot sur un intervalle angulaire de 30 degrés. La figure 2.3 présente les cartes obtenues à différents stades de la simulation, où la flèche rouge représente la pose du robot.

On voit clairement que détecter une cellule libre plusieurs fois diminue à chaque observation la probabilité de son occupation. Plus la couleur vire au noir plus la probabilité de présence d'un obstacle est grande.

7. « occupancy grid maps »

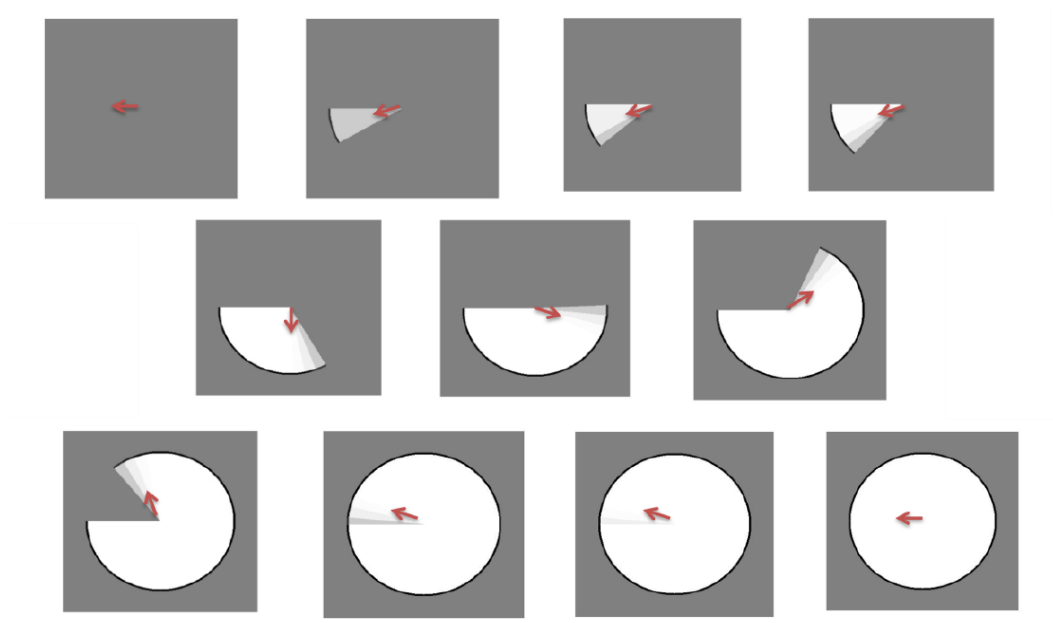

Figure 2.3 Simulation d'une construction avec poses connues d'une carte en grille d'occupation

2.3.3 Filtres Rao-Blackwellized

En cartographie robotique, la complexité et la multitude des sources de perturbations justifie l'utilisation des méthodes de filtrage par la simulation. Le filtrage par la simulation est basé sur un filtre donc un système permettant de reconstruire le signal intéressant (l'état dans un système de navigation) à partir de mesures bruitées acquises au cours du temps. Ce filtre est par contre différent d'un filtre classique comme le filtre de Kalman par le fait qu'il ne cherche pas à trouver la solution optimale au problème d'estimation par analyse directe des observations mais par « la simulation ». En effet, plusieurs états hypothétiques (particules) sont générés selon une loi de probabilité qui modélise le système. À chaque état on associe un poids qui est mis à jour au fil du temps. L'estimation résulte alors de la survie de l'état hypothétique du poids le plus important.

Pour mieux comprendre l'estimation par la simulation regardons cet exemple simple : on considère une carte du monde en noir et blanc et on se propose d'estimer les pourcentages de terre (t) et de mer (m) par rapport à la surface totale[8]. Nous allons donc éviter la lourde tâche de calculer la surface de tous les continents sur la carte. On génére, au lieu, des points (particules) de façon uniformément aléatoire sur toute la surface de l'image de la façon montrée sur la figure 2.4. L'estimation de t est égale au nombre de points blancs sur le nombre total de points. En reprenant l'expérience 5 fois, nous obtenons pour t : 24.78%,

8. Nous savons dèjà qu'ils sont au alentour de $t = 25\%$ et $m = 75\%$. Donc, il s'agit ici de les « retrouver ».

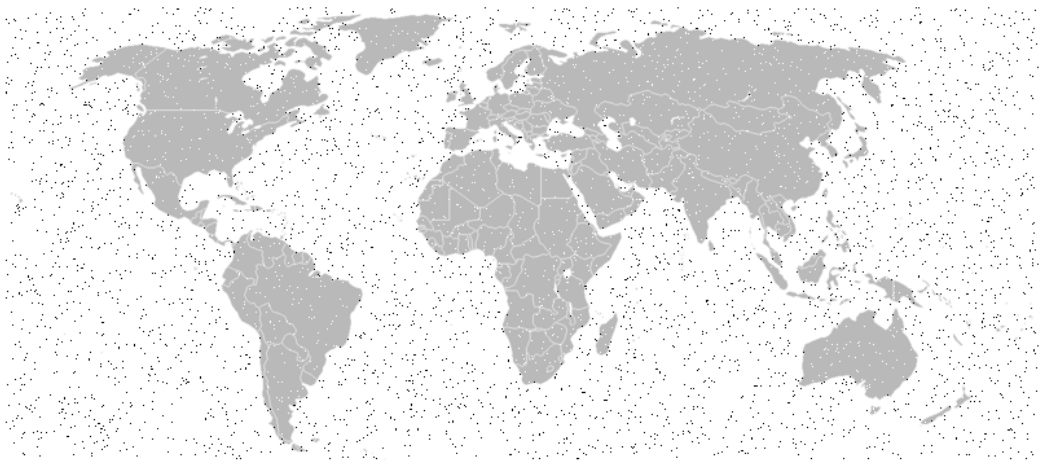

Figure 2.4 5000 points aléatoirement dessinés selon une loi uniforme sur la carte du monde ; les points sur la terre sont blancs, ceux sur les océans sont noirs.

24.8%, 26.48%, 26.26% et 26.52%. La moyenne donne $t = 25.768\%$ et $m = 74.232\%$.

En 1998, Doucet (1998) a introduit le filtre à particules Rao-blackwellized comme méthode efficace de cartographie robotique. Les premiers travaux à étudier la résolution du SLAM en utilisant ce filtre sont Doucet *et al.* (2000) et Murphy (1999). Depuis, d'autres travaux ont permis d'améliorer cette méthode et la rendre plus efficace : Montemerlo (2003), Stachniss *et al.* (2005), Grisetti *et al.* (2005), Grisetti *et al.* (2006), etc. Le *GMapping* de Grisetti *et al.* (2005, 2006) est un algorithme basé sur le filtre Rao-Blackwellized qui a alors vu le jour et qui a connu un grand succès. Aujourd'hui, le *GMapping* est peut être la solution de SLAM la plus utilisée.

Le filtre Rao-Blackwellized est un filtre à particules où l'on cherche à estimer la probabilité $p(x_{1:t}, m|z_{1:t}, u_{1:t-1})$ (toujours avec les mêmes notations que 2.3.1). On fait l'hypothèse que la distribution à estimer peut être factorisée comme suit (« *Rao-Blackwellization* ») :

$$p(x_{1:t}, m|z_{1:t}, u_{1:t-1}) = p(x_{1:t}|z_{1:t}, u_{1:t-1})p(m|z_{1:t}, x_{1:t})$$

Comment justifier cette hypothèse ?
En effet, l'égalité exacte sans aucune approximation est :

$$p(x_{1:t}, m|z_{1:t}, u_{1:t-1}) = p(x_{1:t}|z_{1:t}, u_{1:t-1})p(m|x_{1:t}, z_{1:t}, u_{1:t-1})$$

Le terme $p(m|x_{1:t}, z_{1:t}, u_{1:t-1})$ est remplacé par $p(m|x_{1:t}, z_{1:t})$. On néglige la dépendance en l'odométrie car l'odométrie n'a *apriori* aucun effet sur le modèle de l'environnement surtout que la dépendance en les poses $x_{1:t}$ estimées généralement à l'aide de l'odométrie est plus forte.

Grâce à cette approximation, le problème est dissocié en deux sous-problèmes :

- L'estimation de la trajectoire du robot. Un filtre à particules est un moyen efficace pour résoudre ce sous-problème. Chaque particule est une trajectoire hypothétique avec une carte associée à chaque trajectoire.
- La construction de la carte. C'est le problème classique de construction avec poses connues (voir 2.3.2).

En général, les étapes d'un algorithme de cartographie par filtre *Rao-Blackwellized* sont :

- Prélèvement d'échantillons : Il s'agit de créer une nouvelle génération de particules $\{x_t^{(i)}\}$ de l'ancienne génération $\{x_{t-1}^{(i)}\}$. Le modèle probabiliste de mouvement est communément utilisé pour générer ces échantillons. D'autres échantillonnages plus intelligents sont aussi possibles (Grisetti *et al.* (2006)).
- Mise à jour des poids des particules : Un poids $\omega_t^{(i)} = \frac{p(x_{1:t}^{(i)}|z_{1:t},u_{1:t-1})}{\pi(x_{1:t}^{(i)}|z_{1:t},u_{1:t-1})}$ est accordé à chaque particule (i) pour quantifier son importance (à quel point cette particule mérite-t-elle de survivre ?). π désigne ce qu'on appelle « la distribution de proposition »[9]. C'est une distribution à choisir judicieusement pour diminuer l'appauvrissement en particules ; une situation qui survient quand les particules deviennent « dispersées ». Cette situation survient habituellement au moment de fermeture d'un cycle. La variance augmente drastiquement car très peu de particules arrivent à fermer le cycle correctement. Si on suppose que π vérifie (Doucet (1998)) :

$$\pi(x_{1:t}|z_{1:t}, u_{1:t-1}) = \pi(x_t|x_{1:t-1}, z_{1:t}, u_{1:t-1}).\pi(x_{1:t-1}|z_{1:t-1}, u_{1:t-2})$$

Alors, on peut, en utilisant la règle de Bayes, trouver une relation de récurrence pour la mise à jour du poids :

$$\begin{aligned}
\omega_t^{(i)} &= \frac{p(x_{1:t}^{(i)}|z_{1:t}, u_{1:t-1})}{\pi(x_{1:t}^{(i)}|z_{1:t}, u_{1:t-1})} \\
&= \frac{\eta p(z_t|x_{1:t}^{(i)}, z_{1:t-1}) p(x_t^{(i)}|x_{t-1}^{(i)}, u_{t-1})}{\pi(x_t^{(i)}|x_{1:t-1}^{(i)}, z_{1:t}, u_{1:t-1})} \cdot \underbrace{\frac{p(x_{1:t-1}^{(i)}|z_{1:t-1}, u_{1:t-2})}{\pi(x_{1:t-1}^{(i)}|z_{1:t-1}, u_{1:t-2})}}_{\omega_{t-1}^{(i)}}
\end{aligned}$$

9. « proposal distribution » en anglais

– Mise à jour de la carte : La carte attachée à chaque particule est mise à jour à chaque itération en se basant sur les dernières observations z_t et position $x_t^{(i)}$. Il s'agit d'un problème de cartographie avec poses connues.
– Rééchantillonnage : On effectue le rééchantillonnage en cas de besoin seulement. Il consiste à regénérer de nouvelles particules qui ont toutes le même poids.
D'après Doucet (1998), on rééchantillonne si $N_{eff} = \frac{1}{\sum_{i=1}^{N} (\hat{\omega}^{(i)})^2}$ devient inférieure à un seuil choisi où $\hat{\omega}^{(i)}$ est le poids normalisé de (i). N_{eff} permet de mesurer la dégénérescence de l'algorithme (voir page 12 du rapport technique Doucet (1998)) i.e. l'appauvrissement en particules.

2.4 Aspect probabiliste

« Tous les algorithmes de cartographie robotique ont un aspect commun : ils sont probabilistes » écrit Thrun (2002). Pourquoi les approches probabilistes sont-elles dominantes dans l'histoire de la cartographie robotique ?

En robotique, les erreurs peuvent provenir de plusieurs sources :
– Les changements de l'environnement (voir 2.5.6).
– Les capteurs :
  – à cause de leur portée : à la limite de leur portée la fiabilité des capteurs diminue considérablement.
  – à cause de leur résolution : les capteurs mesurent des grandeurs physiques continues, les contraintes de mémoire imposent alors la quantification des mesures.
  – à cause du bruit : le bruit de mesure est un phénomène inhérent à tout capteur.
  – à cause de leur défectuosité : un capteur peut être défectueux et donc retourner des mesures erronées. Même un capteur en bon état peut connaître un échec de mesure dû à diverses raisons (comme une mauvaise réflexion ou une perte de l'onde réfléchie dans le cas d'un télémètre laser par exemple)
– Les actionneurs : commandes de contrôle exécutées avec des erreurs, glissements, changement des performances à cause de l'usage, etc.
– La programmation :
  – à cause du modèle mathématique : aucun modèle ne simule parfaitement la réalité physique. Il y a toujours des effets négligés, des facteurs non pris en compte pour des raisons de simplification, des approximations dictées par l'intuition...
  – à cause des approximations algorithmiques : une carte, par exemple, représente un espace continu et donc de taille (en tant que structure de données) infinie ! Algorithmiquement, il faut faire des approximations pour représenter cet espace par une

structure de données finie de taille raisonnable.

Le principe des approches probabilistes est de modéliser les sources de bruits et leurs effets sur les mesures et les mouvements. La cartographie nécessite des informations de deux types :

- Des informations de mouvement : pour quantifier le mouvement du robot. L'odométrie par exemple.
- Des informations de perception : pour détecter les obstacles à la portée des capteurs du robot. Des mesures d'un télémètre laser par exemple.

Deux modèles probabilistes au minimum sont alors construits pour simuler le plus fidèlement les mécanismes qui gèrent ces deux types d'informations :

- Un modèle de mouvement : $p(x_t|u_t, x_{t-1})$ où x_t désigne la pose du robot à l'instant t et u_t l'information de mouvement.
- Un modèle de perception : $p(z_t|m, x_t)$ où m désigne le modèle de l'environnement (la carte) et z_t l'information de perception.

Le modèle de mouvement permet de calculer la probabilité de déplacement du robot d'une pose x_{t-1} à une pose x_t étant donné une information de mouvement u_t. En générant un échantillon selon ce modèle sachant que le robot est initalement à la pose x_{t-1} et que l'information de mouvement est u_t nous aurons une prévision qui simule le mouvement du robot avec les bruits et les erreurs impliqués qui sont pris en compte.
De même le modèle de perception permet d'estimer la probabilité d'avoir les mesures z_t à la position x_t du milieu m.

Ces modèles sont indispensables pour deux raisons :

- D'abord, ils représentent les distributions de probabilité intuitives du problème dans le sens où on peut mathématiquement les décrire. En effet, le modèle de mouvement modélise généralement l'odométrie. Ainsi on peut considérer une odométrie parfaite à laquelle on intègre des erreurs angulaires et translationnelles gaussiennes. Et le modèle de perception modélise les capteurs d'obstacles ; les télémètres laser par exemple. On modélise de même les différentes sources de bruit autour de la mesure parfaite.
- La deuxième raison qui justifie l'importance de ces modèles est la méthode d'estimation de l'état (pose + carte). L'estimation se fait par implémentation d'un filtre dont la simplification des lois nous conduit à des relations qui relient ces deux distributions de probabilité. Les deux modèles constituent donc les briques de base de calcul des filtres de la cartographie.

Enfin, il ne faut pas oublier d'évoquer la règle qui est derrière tout filtre utilisé pour la cartographie : la règle de *Bayes*.

Sous sa forme la plus simple, elle s'écrit :

$$p(e|d) = \eta p(d|e)p(e)$$

où e est l'état qu'on cherche à estimer et où d désigne les données acquises.
En cartographie robotique, l'état est habituellement la carte et la pose du robot. Les données sont les informations de mouvement et de perception évoquées plus haut. Donc, dans les algorithmes de SLAM, $p(e|d) = p(x_t, m|z_t, u_t)$. η est le facteur de normalisation. $p(e)$ est la valeur *a priori* de la probabilité de l'état e et $p(d|e)$ est le modèle de génération (modèle de perception par exemple).

2.5 Les difficultés majeures

Les difficultés de la cartographie qui nous intéressent ici sont les problèmes classiques qu'ont historiquement rencontrés les chercheurs. Étant donné que l'histoire de la cartographie robotique est dominée par le recours aux approches probabilistes avec une focalisation sur la résolution du problème de SLAM, il est normal que la majorité de ces difficultés soient étroitement liées à ces pistes de recherche comme, par exemple, le problème de corrélation 2.5.1, la complexité de modélisation 2.5.3, le problème de correspondance 2.5.4, le problème du choix de chemin d'exploration optimal 2.5.5). Du coup, il n'est pas surprenant de voir que la plupart de ces problèmes sont contournés par la stratégie de génération automatique de cartes par interprétation de plans, qui elle a de son côté ses propres difficultés (voir Tableau 2.1).

2.5.1 Problème de corrélation

Ce problème est la difficulté classique qui est au cœur du SLAM au point que certains articles définissent le SLAM par ce problème Grisetti *et al.* (2006).
En effet, on peut le résumer ainsi : pour construire une carte correcte de l'environnement, il faut pouvoir se localiser avec précision par rapport à son milieu (ou sa position initiale). Et pour être en mesure de se localiser avec précision, il faut disposer d'une bonne modélisation de l'entourage du robot. C'est pour cette « dépendence mutuelle entre carte et position du robot » que les chercheurs assimilent le problème de corrélation à celui de l'œuf et la poule [10].
En termes plus liés au registre de la robotique : Pour construire une carte correcte, il faut corriger l'erreur de l'odométrie pour augmenter la précision de la position estimée du robot. Mais en contre partie, pour corriger l'odométrie, il faut avoir une autre source d'information,

10. « chicken egg problem » en anglais

à savoir les données collectées par exploration du milieu, afin de contrer le décalage cumulatif de la localisation.

Si la connaissance des poses du robot est parfaite au cours du temps, la cartographie serait facile par agencement des informations sur les obstacles. L'odométrie est la grandeur mesurée pour quantifier le mouvement total du robot. Comme toute autre grandeur mesurée elle est sujette à du bruit et donc à des erreurs. Mais, dans le cas de l'odométrie le décalage dû aux erreurs de mesures est cumulatif vu que chaque mesure est integrée au mouvement calculé par rapport à la pose de départ. Ce décalage est d'autant plus visible en cas d'erreur angulaire.
Un exemple permet de voir le grand effet de l'erreur angulaire. Imaginons qu'un robot a, en mesurant son odométrie, fait une erreur angulaire e égale à 5 degrés. S'il avance tout droit de $d = 15m$ en supposant qu'il ne fait aucune autre erreur, sa position finale estimée serait à $dec = 2d.\sin(e/2) = 1.3086m$ de la vraie position. Il est du coup trivial de voir l'énorme répercussion sur la construction de carte. Les cartes construites sans correction de la localisation présentent généralement des zones tordues (les couloirs droits sont courbés) et surtout de « graves » inconsistances (désordre, cycles non fermés). Il est possible de voir ces effets sur la figure 9.1 de Thrun *et al.* (2005) par exemple.

D'autre part, si nous avons un modèle de l'environnement, des solutions au problème de localisation sont largement disponibles dans la littérature (Fox (2001), chapitres 7 et 8 de Thrun *et al.* (2005), Montemerlo *et al.* (2002), etc.). Mais, une carte ne peut être construite sans détection des obstacles autour du robot. Les données prises sur ces obstacles sont locales et ne peuvent pas être suffisantes pour assurer la localisation du robot. En effet, dans une pièce carré symétrique par exemple, sans une quantification même approximative du déplacement du robot, ce dernier perd l'orientation facilement. De même, les couloirs symétriques ne donnent pas une indication sur leur sens de parcours.

Ainsi, nous avons à disposition deux sources d'informations qui présentent des faiblesses qui puissent être compensées mutuellement. La maîtrise de cette complémentarité est la clé de la résolution du problème de corrélation.

2.5.2 Complexité en terme de dimension

Les cartes sont des modèles de l'environnement. Du coup, les structures de données utilisées pour les représenter doivent contenir beaucoup d'informations. Vous pouvez essayer de décrire votre bureau ou une pièce de votre maison pour voir le nombre de détails que vous citeriez. Ainsi, nous sommes généralement amenés à consommer énormément de la mémoire en essayant de cartographier l'environnement : cette difficulté limite la tendance à optimiser la modélisation en maximisant les informations retenues. Elle limite également la taille de

la carte. Cette difficulté était plus parlante aux débuts de la cartographie robotique où les ordinateurs de l'époque étaient nettement moins puissants.

La complexité en termes de dimension dépend de plusieurs facteurs :

- la nature de la carte ; grille d'occupation, topologique, etc.
- les dimensions de représentation requises ; 2D ou 3D
- le niveau de détails requis (ou la resolution)

Un exemple concret illustre l'importance de la dimension : une carte modélisant un environnement de $50m$ de largeur par $100m$ de longueur en grille d'occupation 2D nécessite 2.10^6 nombres pour une résolution de $0.05m$.

2.5.3 Complexité de la modélisation

Dans une cartographie robotique dominée par les approches probabilistes (voir 2.2), la modélisation est un défi important.

Généralement, les éléments à modéliser sont :

- La perception : un modèle probabiliste de perception (voir chapitre 6 de Thrun *et al.* (2005)) où l'on tient compte du bruit de mesure, des objets inattendus de l'environnement (comme les passants), des éventuels échecs de mesures, du comportement aléatoire des capteurs etc. est requis. Un tel modèle peut être extrêmement complexe si l'on veut prévoir au maximum les différents bruits inhérents à l'expérience de cartographie.
- Le mouvement : un modèle probabiliste adapté à l'odométrie du robot est requis. En nous inspirant de l'algorithme 5.3 de Thrun *et al.* (2005), nous avons testé un exemple de modèle probabiliste de mouvement avec 1000 échantillons en faisant varier ses paramètres de façon à équilibrer les erreurs angulaire et translationnelle dans une première expérience, avoir une erreur translationnelle prépondérante dans la deuxième et une erreur angulaire prépondérante dans la troixième. Les nuages de points obtenus sont donnés par la figure 2.5. Le choix des paramètres adapté à l'odométrie d'un robot spécifique n'est pas évident.

2.5.4 Problème de correspondance

Connu aussi sous le nom du « problème d'association de données »[11], c'est peut-être le problème le plus difficile de la cartographie robotique selon Thrun (2002). Au débuts de la recherche, il a été ignoré jusqu'à la fin des années 90.

Il s'agit du problème de reconnaissance du même endroit par lequel le robot est passé une première fois. Il est rencontré dans les environnements se présentant sous la forme d'un

11. « *data association problem* » en anglais

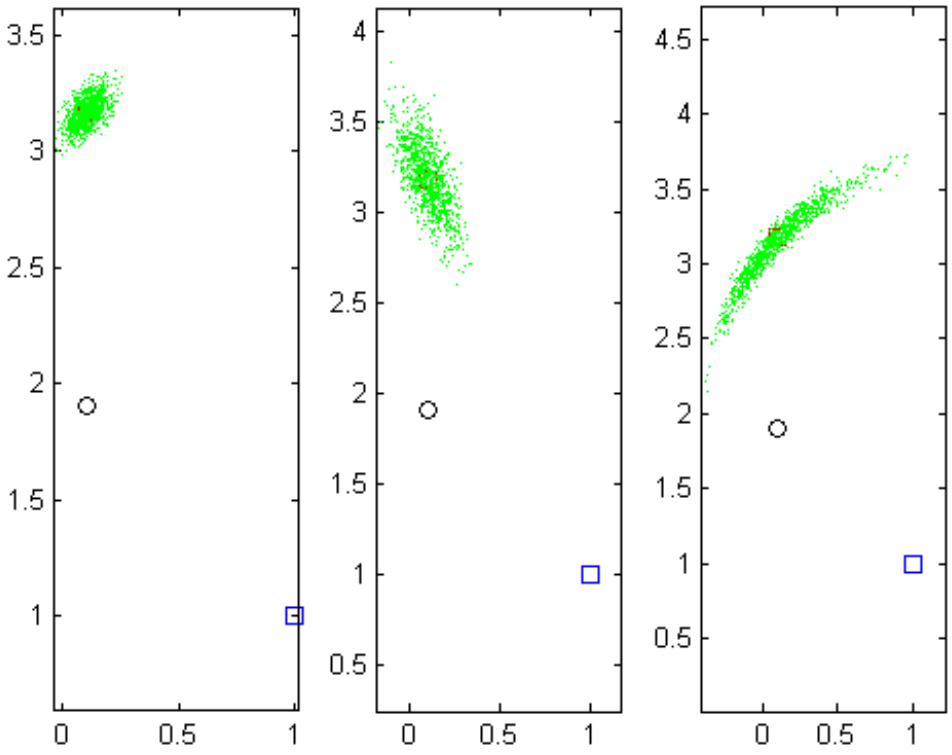

Figure 2.5 Nuage de point de la simulation du mouvement élémentaire d'un robot pour 1000 échantillons avec différents paramètres de bruit (rectangle bleu : pose de départ, rectangle rouge : pose d'arrivée, cercle noir : centre de l'arc du mouvement élémentaire)

cycle suffisamment large pour que le robot ne détecte aucun obstacle de la zone de fermeture du cycle hors de cette zone le long du chemin d'exploration. Plus le cycle est large, plus le problème est difficile à résoudre. En effet, comme nous l'avons précisé en 2.5.1, l'erreur d'odométrie est cumulative. En retrouvant la zone de fermeture du cycle, le robot pourrait alors être complètement en décalage dans sa carte et du coup, la reconnaissance de la zone lui est impossible. Dans ce cas, les cartes construites sont du genre de la figure 2.6.

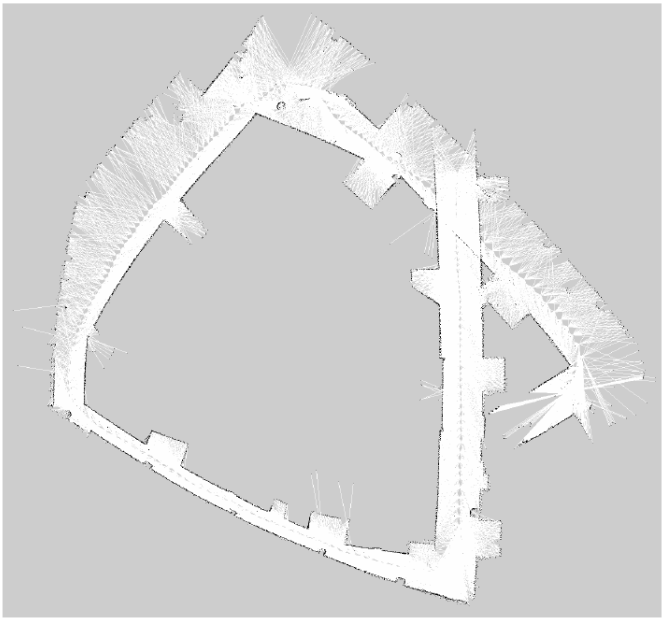

Figure 2.6 Cartographie de l'étage 1 du pavillon Lassonde de l'École Polytechnique avec mauvaise gestion du problème de correspondance : Carte prise de El-Fathi (2012) sous permission de l'auteur

2.5.5 Chemin d'exploration

Les environnements à cartographier par exploration sont normalement inconnus pour le robot. Ainsi, une stratégie d'exploration est nécessaire aux milieux où le robot est livré à lui même pour explorer automatiquement sa carte. Tel est le cas des robots envoyés pour explorer les planètes ou des robots militaires conçus pour l'espionnage.

Cette difficulté est intuitive si l'on essaie d'imaginer la situation où l'on débarque pour la première fois dans un environnement inconnu sans carte. La navigation est certes beaucoup plus facile et « économique » dans le cas où l'environnement visité est complètement modélisé.

Plusieurs critères rentrent dans la planification du chemin d'exploration :

– Intérêts d'exploration : Qu'est qu'on cherche à trouver ? Une carte détaillée de tous les objets physiques est-elle nécessaire ?

- Gestion des imprévus et planification de la fin d'exploration : Quel changement de plan à prévoir en cas d'imprévu ? Où aller si objectifs atteints ? Comment retourner à « la base » à la fin de la mission d'exploration ?
- Classification des priorités d'exploration
- Temps
- Énergie
- autres critères...

Une planification intelligente du chemin d'exploration qui tient compte de tout ces critères est à elle seule un problème très complexe.

2.5.6 Changements de l'environnement

Les environnements à cartographier ne sont pas toujours statiques. Ils connaissent diverses changements au cours du temps. Ces changements peuvent être lents [12] (le déplacement d'une poubelle, l'ouverture d'une porte, le déplacement d'une chaise...) comme ils peuvent être rapides (des piètons qui passent...). Tous ces changements sont difficiles à gérer dans la cartographie pour plusieurs raisons :

- Les obstacles dynamiques qui changent de position rapidement causeront, si pris en compte dans la carte construite, l'instabilité de la carte et laisseront forcément des traces dans le résultat obtenu à l'issue de la cartographie. C'est pour cette raison que ces obstacles sont assimilés grâce aux approches probabilistes utilisées à du bruit.
- Assimiler les changements rapides à du bruit pose également d'autres difficultés. En effet, ce bruit est modélisé au sein du modèle de perception qui cherche à simuler le fonctionnement des capteurs du robot. On utilise généralement une distribution exponentielle pour décrire cet effet (voir pages 155-156 de Thrun *et al.* (2005)). On suppose donc que l'apparition des obstacles dynamiques dans le champs des capteurs est aléatoire et inattendue. Dans le cas où l'objet dynamique suit le robot à la même position relative durant le SLAM (comme par exemple, une personne qui suit le robot de près), cette hypothèse n'est plus vérifiée.
- Les changements lents sont, d'un autre côté, problématiques aussi. Imaginez le robot face à une porte fermée qui est dans sa carte ouverte car au moment de la construction de la carte elle était effectivement bien ouverte. Là deux hypothèses sont plausibles pour le robot : soit l'état de la porte a changé, soit le robot ne s'est pas localisé correctement. Laquelle entre les deux choisir et comment ? Un raisonnement intelligent devrait être prévu et programmé pour gérer ce genre de situations.

12. « lents » dans le sens où ils sont moins fréquent dans le temps que les changements « rapides »

2.5.7 Obstacles transparents

Les télémètres laser sont des capteurs qui permettent de mesurer la distance aux obstacles par lancement de rayons laser. Le déphasage entre le rayon émis et le rayon réfléchi permet de calculer cette distance.

L'utilisation de ces capteurs pour la détection des obstacles est devenue de plus en plus fréquente sur les robots au dépend de l'utilisation des télémètres ultrason beaucoup moins performants car beaucoup moins précis et de portée moins élevée. Aujourd'hui, ce sont les télémètres laser qui dominent en robotique. En effet, les algorithmes de cartographie récents ont des modèles de perception conçus spécialement pour exploiter des données laser. *Le GMapping* de Grisetti *et al.* (2006) en est un exemple. Comme le principe de fonctionnement de ces capteurs est axé autour de la réflexion d'une onde lumineuse sur les surfaces des objets détectés, les obstacles transparents sont alors problématiques à cause du phénomène de réfraction de la lumière. Ces obstacles sont difficiles à détecter par les lasers. Les cartes construites risquent alors d'être « dangereusement » incomplètes.

El-Fathi (2012) a rencontré ce problème et a pu expérimenter la cartographie en présence des obstacles transparents en utilisant *Le GMapping*. La figure 2.7 illustre le problème dans une carte qu'il a obtenue en milieu contrôlé. On voit bien que les obstacles transparents sont manquants sur la carte ce qui est dangereux pour la navigation. El-Fathi propose une tentative de solution à ce problème. Son idée consiste à transformer les données sonar de façon à les adapter à l'entrée du module de cartographie. Les résultats obtenus dans ce même milieu (Figure 2.8) ainsi que d'autres tests exposés par El-Fathi prouvent le manque d'efficacité de cette méthode.

Bien évidemment, il existe des solutions de SLAM qui ont permis de résoudre le problème des obstacles transparents par fusion des données laser et sonar (Diosi et Kleeman (2004); Diosi *et al.* (2005)), cependant il serait intéressant d'avoir, par une approche modulaire, une solution (qui puisse être généralisée dans un second temps) et qui permet une correction des cartes retournées par les modules de cartographie conçus pour les lasers. Le *GMapping*, en particulier, est très utilisé et a fait preuve d'efficacité, il serait alors dommage de l'abondonner dans certaines situations où il y a des obstacles transparents à cartographier.

2.6 Conclusion et pistes récentes

Les méthodes connues et les plus développées dans l'histoire de la cartographie robotique sont presque toutes focalisées sur la stratégie par exploration. Le recours au calcul probabiliste a également dominé la recherche dans ce domaine. L'approche par filtre Rao-Blackwellized, en particulier *le GMapping*, représente selon nous un exemple remarquable et représentatif

Figure 2.7 Cartographie d'un milieu contrôlé en présence d'obstacles transparents (représentés en rouge) en utilisant les télémètres laser seulement : Carte prise de El-Fathi (2012) sous permission de l'auteur

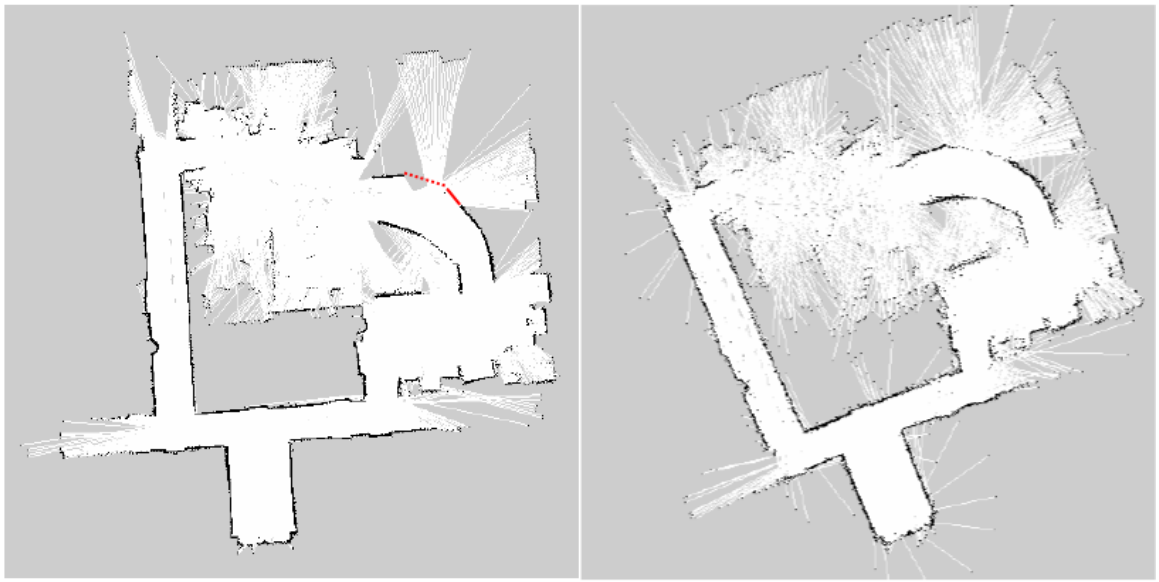

Figure 2.8 Cartographie d'un milieu contrôlé en présence d'obstacles transparents (représentés en rouge) en utilisant les télémètres laser et ultrason (carte à droite) : Cartes prise de El-Fathi (2012) sous permission de l'auteur

de l'exploitation de la probabilité pour résourdre le problème de SLAM. *Le GMapping* exploite aussi la méthode des grilles d'occupation qui a permis une représentation efficace des environnements. Il faut également noter que le problème de correspondance est assez bien géré dans beaucoup de situations par cet algorithme très utilisé.

Cependant, avec tout ce qui a été réalisé certaines pistes restent encore à explorer et des méthodes récentes continuent à apparaître pour traiter certains problèmes qui ont persisté jusqu'à aujourd'hui.

La stratégie de cartographie par exploitation des plans architecturaux a déjà attiré l'attention des chercheurs (Schafer *et al.* (2011); Farhan *et al.* (2011)). Nous notons le recours à l'idée directe par les outils de traitement d'image, ainsi que l'exploitation plus intelligente du format DXF, mais nous n'avons pas encore un algorithme qui permet de convertir un plan en une grille d'occupation. Le travail de Schafer *et al.* constitue un début pour lancer l'exploration de cette piste de recherche dans l'objectif de bâtir le premier algorithme de construction sans exploration des grilles d'occupation.

Le problème des changements de l'environnement a été contourné, pour la plupart des travaux, en supposant que le milieu à cartographier est statique. Les objets dynamiques sont alors assimilés à du bruit pris en compte dans les modèles de perception. Cette approche ne tient habituellement pas compte des fréquences de ces changements ce qui entraîne généralement la non représentation sur la carte construite des obstacles bougeant à « hautes fréquences » (les passants par exemple), ce qui est correcte généralement. Cependant, les changements plus « lents » (état d'une porte par exemple) sont mal gérés par les méthodes classiques. La nécessité de trouver une solution à ce problème trouve alors son importance dans le cas des robots qui opèrent sur une longue période de temps. Pour ce cas, des travaux très récents proposent une idée de solution (Krajnik *et al.* (2014); Duckett et Krajnik (2014)). L'idée consiste en l'analyse fréquentielle des activités qui causent les changements de l'environnement afin de prévoir ces changements. Comme les environnements humains (milieux intérieurs) connaissent des changements répétitifs et généralement organisés selon les habitudes des activités humaines qui ont lieu dans le milieu, Duckett et Krajnik ont pu modéliser les processus qui changent la structure de l'environnement. Les changements sont du coup étudiés, prévus et gérés selon leurs fréquences afin d'assurer une construction et une mise à jour correctes de la carte.

CHAPITRE 3

ENRICHISSEMENT INTELLIGENT PAR DONNÉES SONAR DES CARTES GÉNÉRÉES PAR DÉTECTIONS LASER VIA SLAM

Ce chapitre est consacré à la présentation d'une première technique d'optimisation de la navigation du FRMI. Cette technique permet de proposer une solution aux limites de la cartographie par SLAM disponible (voir El-Fathi (2012)), liées à l'utilisation inadaptée de capteurs de caractéristiques différentes comme entrées à un module conçu pour les détections laser. Nous présenterons ces limites en une première partie 3.1. Puis, nous en déduirons, dans une seconde partie 3.2, une description détaillée de l'architecture de la solution envisagée. La troisième partie 3.3 décrira l'algorithme proposé, puis, les différentes méthodes d'enrichissement simulées sont exposées à la partie 3.4 pour en aboutir à un choix justifié de la(les) méthode(s) la(les) plus adaptée(s) au FRMI. Enfin, nous présenterons le programme final implémenté sur le fauteuil et les résultats qui en résultent dans diverses situations en 3.5 et 3.6.

3.1 Limites de la cartographie disponible

Grâce aux travaux de El-Fathi (2012) et Boucher (2010), le FRMI est équipé de capteurs (Encodeurs pour mesures d'odométrie, 3 télémètres Laser, 6 sonars et caméra Kinect) (voir El-Fathi, 2012, sect. 3.2) et d'une architecture de contrôle (voir El-Fathi, 2012, sect. 4). Celle-ci intègre un ensemble de modules permettant l'automatisation de nombreuses tâches, locales : traversée de porte, évitement d'obstacles... et globales : navigation point à point et, localisation et cartographie simultanée par acquisition de données laser ou bien par fusion des mesures de la Kinect, des lasers et des sonars en une structure de données jouant le rôle de nuage de points (voir El-Fathi, 2012, sect. 3.6) analogue à celui retourné par les lasers.
La solution de fusion des mesures n'a pas donné des résultats satisfaisants tel que présenté à la sect. 5.1.1. de El-Fathi (2012). La solution ne permet pas de représenter efficacement les obstacles transparents. El-Fathi explique ceci par des notions probabilistes générales. Etudions avec plus de profondeur les limites de cette architecture.

En effet, les modules implémentés sur le FRMI par El-Fathi se basent sur un algorithme de cartographie et localisation simultanées (SLAM) ; le *GMapping* qui n'utilise qu'un modèle de filtre adapté aux seules données des télémètres laser, comme la plupart des modules de SLAM récents qui génèrent la carte sous format de grille d'occupation (Le *GMapping* est peut-être

l'algorithme le plus utilisé parmi cette famille aujourd'hui). Ces modules présentent ainsi le défaut majeur de la faible performance de représentation de certains types d'obstacles. Les obstacles transparents, comme notamment les vitres que l'on trouve abondamment dans plusieurs milieux intérieurs visités par la communauté des utilisateurs de fauteuils roulants (par exemple, les centres commerciaux, les centres de réadaptation, etc.) sont difficiles à détecter avec des capteurs optiques comme les télémètres lasers ou la caméra Kinect.

On pourrait penser que recourir à un module de cartographie se basant sur les détections sonars, vu que les sonars détéctent les obstacles transparents, représente une bonne alternative.
Certes, l'utilisation exclusive des sonars avec un module de cartographie adapté permettra d'assurer la représentation de tous les objets rencontrés quels que soient leurs transparences. Il y a eu déjà beaucoup de travaux qui s'intéressent à la cartographie par données sonars (Moravec et Elfes (1984); Elfes (1987); Ribo et Pinz (2001); Gambino *et al.* (1996)) à l'époque où ces capteurs traditionnels suscitaient plus d'intérêt dans le domaine de la cartographie robotique. Cependant, les capteurs ultrason étant des détecteurs d'obstacles de faibles performances en comparaison aux télémètres laser, ainsi que la connaissance que nous avons de la rareté des milieux dominées par les obstacles transparents (on peut rencontrer une pièce, une vitrine ou même une façade en vitre mais très rarement tout un bâtiment transparent !) nous incite à éviter d'abandonner des capteurs extrêmement précis et d'une meilleure performance globale de détection, et donc d'une meilleure efficacité de correction des erreurs cumulatives de l'odométrie, comme les lasers.
Mais nous cherchons ici à trouver une technique qui nous permettrait de profiter de la capacité des sonars à détecter ces obstacles particuliers et donc de les représenter avec efficacité sans pour autant dégrader la performance atteinte en construction et en localisation avec les algorithmes de SLAM récents.

L'objectif principal de ce chapitre est alors de proposer une solution générale permettant de remédier au défaut présenté ci-dessus. Mais, présentons d'abord le système de navigation que nous visons à améliorer et l'algorithme de SLAM disponible pour mieux mettre en évidence ses limites.

3.1.1 Architecture du système à optimiser

Capteurs

Afin de créer une carte tout en localisant le robot (objectif d'un algorithme de SLAM), il est intuitif qu'il faut recevoir des informations sur l'environnement du robot, notamment des obstacles statiques qui y sont présents. Il est également évident qu'il est nécessaire de quanti-

fier au fil du temps le mouvement du robot pour pouvoir le localiser dans cet environnement. Ainsi, nous avons besoin de deux types de données :

- **Données de perception :** que nous allons noter z_t dans la suite. Ces derniers traduisent « la perception » du robot de son environnement extérieur. Pour le fauteuil, qui est ici notre robot, nous disposons de 3 types de capteurs pouvant retourner ces données qui sont les télémètres laser, les sonars et la Kinect (nous allons focaliser sur les deux premiers capteurs dans notre travail) ;
- **Données de mouvement :** que nous allons noter u_t dans la suite. Ces derniers permettent de donner une idée globale sur le mouvement de robot au fil du temps par le calcul de l'odométrie. Ils sont retournés par les encodeurs optiques rotatifs liés aux roues motrices.

Les capteurs de notre système de navigation sont ainsi introduits et nous pouvons les classer selon leur mode de fonctionnement en deux types :

Capteurs extéroceptif : « qui se basent sur des mesures prises [par le robot] par rapport à son environnement global »[1].

- **Télémètres laser :** Il s'agit d'un dispositif permettant de mesurer le déphasage entre le rayon laser émis et reçu afin d'en déduire la distance parcourue et donc celle qui sépare le capteur à l'obstacle perçu (le premier rencontré par le rayon). Ce capteur se caractérise par une bonne portée et une grande précision El-Fathi (2012). Celui que nous utilisons sur le FRMI est un *Hokuyo UHG-08LX* (au nombre de trois) ayant les caractéristiques suivantes :
  - Angle de vue de 270 degrés
  - Résolution angulaire de 0.36 degré
  - Fréquence de balayage de 40Hz
- **Sonars :** Ils ont le même mode de fonctionnement que les télémètres laser à la différence de la nature de l'onde utilisée. Ce capteur utilise, en effet, une onde ultrason. Il est nettement moins précis et de portée moins large que les lasers. La nature de l'onde, se représentant le mieux sous forme de cône qui s'élargit en s'éloignant du robot, justifie en partie cette précision et cette portée moins bonnes par rapport au laser dont les rayons restent assez parallèles à l'échelle des environnements visités.

Capteurs proprioceptif : « qui effectuent leurs mesures par rapport à ce qu'ils perçoivent localement du déplacement du robot »[2].

- **Codeurs optiques rotatifs :** Ces capteurs assurent le comptage du nombre de tours de chaque roue à intervalles de temps connus. Ces nombres permettent alors de calculer

1. définition de wikipédia
2. définition de wikipédia

l'odométrie donnée par $[\nu, \omega]^T$ avec :
$\nu = \frac{\pi R}{n}(N_{droite} + N_{gauche})$ et $\omega = \frac{2\pi R}{nL}(N_{droite} - N_{gauche})$
où n est la résolution des encodeurs égale à 110 tiques par rotation, les N sont les nombres de tours comptés (qui sont remis à zéro à chaque mesure) et L la distance entre les deux roues.

Nous concluons alors deux points qui nous intéressent :

– la large différence de performances entre sonars et lasers. D'où, nous pouvons déduire que les deux capteurs ne devraient pas donner des résultats de cartographie satisfaisants d'un milieu contenant des obstacles transparents s'ils ne sont pas modélisés par deux modèles de perception différents au sein du même module.
– l'erreur d'odométrie est cumulative ce qui nécessite une correction de la position par exploitation des détections les plus fiables (rôle du *scan matching*[3] et de la partie de localisation du filtre du module de SLAM. Voir 3.1.2)

Description structurelle

Par la figure 3.1, nous avons représenté la structure du système tel qu'il a été réalisé par El-Fathi. Nous voyons bien que la fusion des mesures des différents capteur se fait à l'extérieur du module de SLAM qui est alors considéré comme une boite noire à *inputs* et *outputs*.

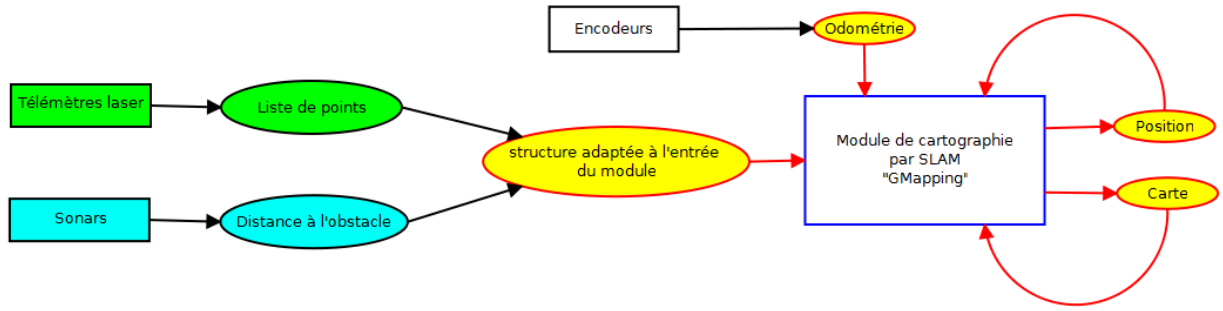

Figure 3.1 Architecture de la cartographie à optimiser

Notre idée consiste à traiter chaque type de mesure de façon spécifique et adaptée, en favorisant intelligemment, éventuellement, un type sur l'autre en localisation et en assurant une bonne précision de la carte tracée. L'architecture de notre système devrait alors avoir la structure de la figure 3.2.

3. Technique de correction de la position par calcul de la transformation entre deux ensembles de détections laser décalés dans le temps

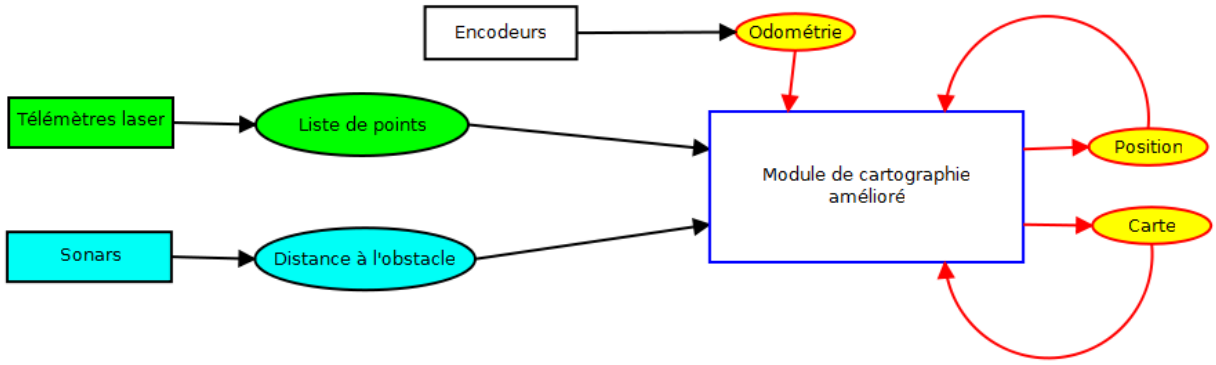

Figure 3.2 Architecture proposée de la cartographie améliorée

Cette structure suppose que l'on ne considère plus le module de cartographie comme une boite noire vu que le module orignal verra une modification que nous privilégierons qu'elle soit par augmentation par l'ajout d'une couche supplémentaire pour le traitement des mesures des sonars.

3.1.2 Algorithme *GMapping*

Ici nous ouvrons « la boite noire » du module de SLAM à optimiser, évoquée ci-dessus, pour mieux comprendre le fonctionnement du module de cartographie de l'intérieur. Cette compréhension justifiera en partie les choix qui ont abouti à notre technique d'optimisation comme nous allons l'expliquer.
Le *GMapping* est basé sur un filtre particulaire (voir 2.5.3) qui profite de la *Rao-blackwellization* (voir 2.3.3) pour dissocier le problème du SLAM en deux sous-problèmes distincts : **L'estimation de la trajectoire du robot** et **la construction de la carte**.

$$p(x_{1:t}, m|z_{1:t}, u_{1:t-1}) = \underbrace{p(x_{1:t}|z_{1:t}, u_{1:t-1})}_{\substack{estimation \\ de\ la\ trajectoire}} \underbrace{p(m|z_{1:t}, x_{1:t})}_{\substack{construction \\ de\ la\ carte}}$$

où m désigne la carte, et, $x_{1:t}$ les positions du robot jusqu'à l'instant t.
Le détail de l'architecture montrant les composants internes du module est donné à la figure 3.3.

Localisation

La construction de la carte dépend de la localisation car elle se sert de la position estimée pour effectuer correctement la mise à jour de la carte. Ainsi, la précision de la localisation

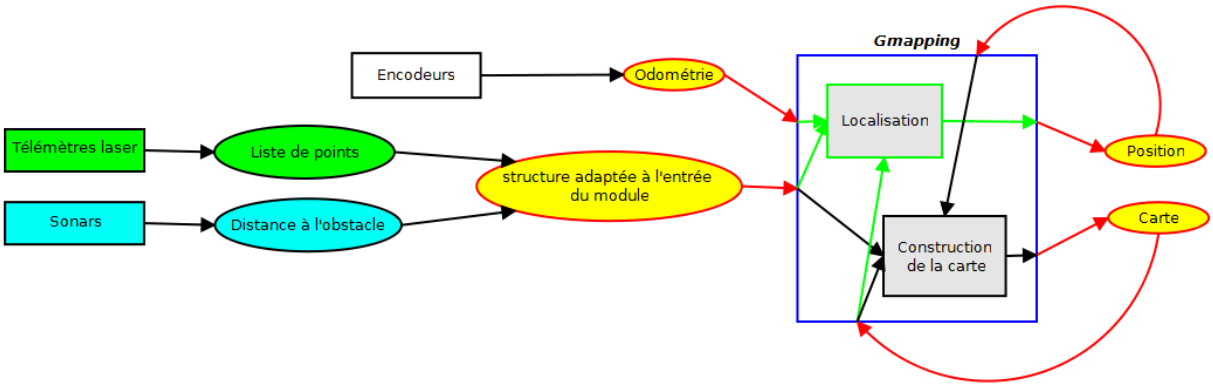

Figure 3.3 Les composants majeurs du *GMapping*

est d'une importance indiscutable pour assurer l'obtention d'une carte où l'erreur cumulative de calcul de l'odométrie ne se répercute pas en une grille d'occupation inexploitable. En particulier, en milieux d'une certaine complexité [4] le robot pourrait avoir à fermer une boucle et donc à reconnaitre un endroit par où il est passé précédemment. Dans ce cas, si le décalage de la position estimée est important, le robot ne ferme pas la boucle convenablement car il ne cherche par à reconnaitre la bonne zone de la carte en cours de construction.

D'après Grisetti *et al.*, la localisation du GMapping s'effectue sur chaque particule de l'échantillon de particules selon les étapes suivantes :

- D'abord, une extrapolation de la position à l'aide de la mesure d'odométrie. Puis, un *Scan Matching* qui permet de déterminer un mode (une première estimation autour de laquelle on échantilllonne) à partir de la position estimée et de la carte construite à l'itération précédente ainsi que les mesures (de laser normalement) prise à l'instant t du traitement en cours.
- Echantillonnage autour du mode et estimation de la distribution de proposition gaussienne à l'aide de ces positions générées.
- Génération de la pose de la particule à l'instant t selon la distribution gaussienne et mise à jour de son poids.

Le *Scan Matching* et le calcul de la distribution de proposition font intervenir une probabilité importante qu'il est indispensable d'être en mesure d'estimer correctement. Il s'agit de la probabilité de l'obtention d'une observation z_t à un état x_t du robot étant donnée une « représentation » donnée de l'environnement (notre carte) m. Cette probabilité représente le modèle de perception du capteur utilisé. Grisetti *et al.* précisent dans leur article qu'ils

4. parmi les critères classiques qui traduisent la complexité d'un milieu donnée ; les boucles dans l'environnement à cartographier, celles-ci représentent un défi important.

« utilisent 'le modèle de l'extrémité du rayon' [5] pour calculer » cette probabilité.
Pour ce modèle (voir Thrun *et al.*, 2005, sect. 6.4), les rayons sont considérés indépendants, approximation qui n'est pas du tout acceptable dans le cas de l'assimilation d'un cône de perception d'un sonar à un ensemble de rayons (ce qui est justement fait en conditionnant les mesures du sonar à l'entrée du module). De plus, les extrémités des rayons sont assimilées à des points et la probabilité est calculée en fonction de la distance des obstacles à ce point de détection. Tandis que dans le cas des sonars, tout un ensemble de points formant un arc devrait être associé à une détection d'un point-obstacle quelque part sur l'arc.

Ainsi, nous pouvons conclure que la solution de fusion externe des mesures des lasers et celles des sonars n'aide en rien à améliorer la localisation. Nous décidons alors dans notre solution de nous restreindre à l'utilisation des lasers pour assurer la localisation. Les données sonars vont ainsi nous sevir exclusivement à l'enrichissement de la carte (la position étant connue car estimée par la localisation du *GMapping* avec les détections laser)

Construction de la carte

L'objectif est de représenter l'environnement du robot par une carte composée de grilles auxquelles on attache à chacune une probabilité d'occupation : $p(m_i|z_{1:t}, x_{1:t})$. Pour simplifier le problème, on fait l'hypothèse que les probabilités d'occupation des cellules sont indépendantes les unes des autres, donc, $p(m|z_{1:t}, x_{1:t}) = \Pi p(m_i|z_{1:t}, x_{1:t})$. Le théorème de Bayes conduit ensuite à établir une relation qui permet de trouver un algorithme de calcul itératif.

$$\frac{p(m_i|z_{1:t}, x_{1:t})}{1 - p(m_i|z_{1:t}, x_{1:t})} = \frac{p(m_i|z_{1:t-1}, x_{1:t-1})}{1 - p(m_i|z_{1:t-1}, x_{1:t-1})} \cdot \frac{p(m_i|z_t, x_t)}{1 - p(m_i|z_t, x_t)} \cdot \frac{1 - p(m_i)}{p(m_i)}$$

$p(m_i)$ étant la probabilité préalable à l'occupation. Dans cette expression, la probabilité clé est $p(m_i|z_t, x_t)$. Il s'agit de la probabilité associée à la cellule i quand le robot est à la position x_t et qu'il a pris les mesures z_t. C'est en quelque sorte un modèle inverse de perception.
Un modèle inverse des lasers, tout comme un modèle de perception, dépend de la nature des mesures, des performances et donc des considérations d'approximation relatées au paragraphe ci-dessus. Dans un milieu contenant des obstacles transparents, l'obtention d'une carte précise où figurent tous les obstacles opaques rencontrés en n'utilisant que les lasers est testé (voir El-Fathi, 2012, sect. 5.) et fonctionne efficacement. Nous nous proposons, maintenant que l'architecture complète de notre solution se clarifie, de mettre au point un composant d'enrichissement intelligent de la carte construite par le *GMapping* faisant intervenir un modèle inverse des sonars à l'image du composant de construction de carte décrit ici.

5. ma traduction du « beam endpoint model »

3.2 Architecture de solution

L'architecture de la cartographie optimisée que nous proposons a été construite en veillant à satisfaire certains points pratiques et techniques.

D'une part, nous avons un algorithme testé et largement utilisé, et donc, qui a fait preuve de robustesse et de performance assurées, en particulier sur notre FRMI en utilisant comme entrées les seules données lasers. Son défaut visé par la technique présentée ici est la non représentation des obstacles transparents. Ainsi, nous avons décidé de le garder intact ce qui nous épargne la lourde et délicate tâche de nous plonger dans son code original en vu de le modifier de l'intérieur. La modification serait ainsi externe par l'ajout d'une couche qui joue sur les outputs du *GMapping* pour enrichir la carte-laser avec les mesures des sonars.

D'autre part, un enrichissement intelligent nous permettant de profiter de la représentation des espaces libres explorés par les rayons laser derrière les vitres valorise notre choix d'architecture par rapport à une solution par fusion adaptée (au moyen de bons modèles de perception) des mesures des capteurs en amont de la construction de la carte.

La figure 3.4 représente les composants et les différents flux de données de la cartographie améliorée que nous proposons.

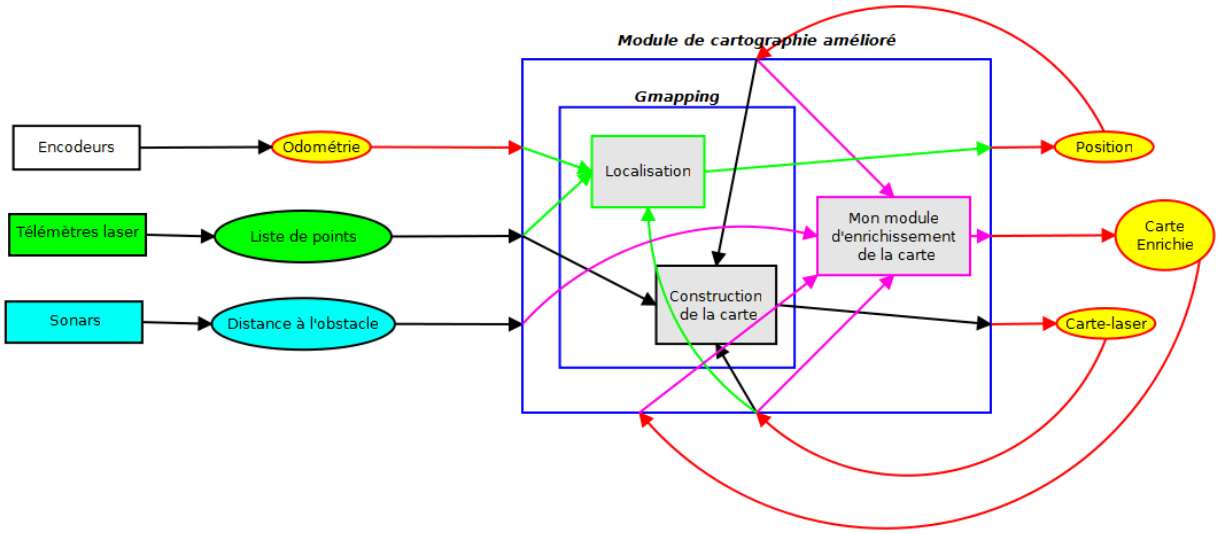

Figure 3.4 Architecture de solution

3.3 Algorithme solution

3.3.1 Entrées et Sortie

Les entrées du composant sont :

- Les mesures des sonars qui sont des réels positifs représentant la distance du sonar correspondant à l'obstacle détecté à l'instant t.
- Les poses des sonars dans le référentiel lié à la carte (ou référentiel *monde*) à l'instant t. Chaque pose est bien évidemment obtenue à partir de la position du robot dans la carte-laser, estimée par le *GMapping* et connaissant la pose de chaque sonar par rapport au châssis du robot (chaque sonar est fixé par une liaison rigide sur le corps du FRMI)
- La carte la plus récente retournée par le *GMapping* construite en utilisant les détections des lasers uniquement. Nous appelerons ces cartes : cartes-laser. Ces cartes fournissent en plus de la grille de cellules l'ensemble des informations importantes liée à la carte telles que ses dimensions et surtout la position de sa cellule-origine $(0, 0)$ dans le repère *Monde*.
- La carte-sonar [6] à l'instant $t - 1$.

La seule sortie est la carte enrichie par les mesures des sonars.

3.3.2 Étapes de l'algorithme

Le composant permet de mettre à jour la carte-sonar à chaque itération, à chaque réception des mesures des sonars.

Au sein de chaque itération, les traitements sont les suivants :

- D'abord, on initialise une grille à cellules de même taille que celle de la carte-laser. Dans cette grille, on marque les cellules appartenant à un obstacle et les cellules d'occupations inconnues de telle sorte à faciliter leur différentiacion entre elles et leur différentiacion des cellules libres.
- Ensuite, on choisit la zone de traitement afin d'éviter de traiter toute la carte. Ce choix se fait tout simplement en prenant le max et le min des abscisses et le max et le min des ordonnées des positions des sonars et d'en définir un rectangle tel que montré sur la figure 3.5. Bien évidemment, les grandeurs délimitant cette zone sont converties en nombre de cellules au lieu des mètres.

  Le pseudo-code résumant cette étape est présenté à l'Algorithme 1.
- Puis, on agrandit en cas de besoin la carte-sonar $t - 1$ pour l'adapter à la taille de la carte-laser t et on remplit la grille de cellules avec les valeurs a priori d'occupation

6. Nous allons appeler **carte-sonar** la carte enrichie de format adapté à la méthode de calcul probabiliste utilisée. Cette carte a un format différent de celui de la **carte-laser** qui est retournée par le *GMapping*.

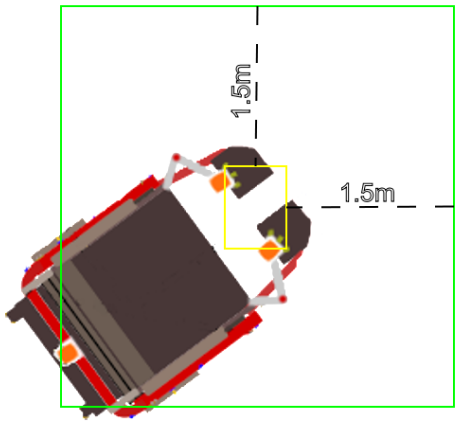

Figure 3.5 Figure explicative de la détermination de la zone de traitement où $1.5m$ représente la portée maximale de nos sonars

provenant de la carte-sonar agrandie.

- A ce niveau-là, nous aurions préparé ce qu'il faut pour la boucle d'enrichissement qui est le cœur de l'algorithme. Au sein de cette boucle, on accède à la mesure et aux informations de pose de chaque sonar et on parcourt les cellules de la zone de traitement de la grille de cellules en effectuant les traitements suivants pour chaque cellule n'appartenant pas à l'ensemble d'éléments d'états inconnus ou occupés (voir Algorithme 2) :
  - On détermine les coordonnées polaires de la cellule par rapport au télémètre ultrason, à savoir la distance ρ de la cellule au centre du sonar ainsi que son angle ϕ par rapport à l'axe des abscisses de celui-ci (voir Figure 3.6). Soit (x, y) les coordonnées de la cellule dans le repère de la carte. x et y sont obtenus à partir de la position (i, j) de la cellule dans la grille :

$$\begin{aligned} x &= (i - 0.5)res + o_x \\ y &= (j - 0.5)res + o_y \end{aligned}$$

où $res = carte_{laser}.resolution$, $o_x = carte_{laser}.origine.x$ et $o_y = carte_{laser}.origine.y$. Ainsi, connaissant la position (x_s, y_s) et l'orientation yaw_s du sonar dans le repère

Algorithm 1 Algorithme de choix de la zone de traitement

Initialisation des limites
$x_m = carte_{laser}.largeur$
$x_M = 0$
$y_m = carte_{laser}.hauteur$
$y_M = 0$
Boucle de choix de la zone de traitement
pour $i = 1...nbSonars$ **faire**
$x_m = min(x_m, arrondi(\frac{sonar(i).pose.x - 1.5 - carte_{laser}.origine.x}{carte_{laser}.resolution}))$
$x_M = max(x_M, arrondi(\frac{sonar(i).pose.x + 1.5 - carte_{laser}.origine.x}{carte_{laser}.resolution}))$
$y_m = min(y_m, arrondi(\frac{sonar(i).pose.y - 1.5 - carte_{laser}.origine.y}{carte_{laser}.resolution}))$
$y_M = max(y_M, arrondi(\frac{sonar(i).pose.y + 1.5 - carte_{laser}.origine.y}{carte_{laser}.resolution}))$
fin pour
Ajustement selon les dimensions de la carte
$i_{init} = max(0, x_m)$
$i_{end} = min(carte_{laser}.largeur, x_M)$
$j_{init} = max(0, y_m)$
$j_{end} = min(carte_{laser}.hauteur, y_M)$

de la carte, on déduit alors ρ et ϕ :

$$\begin{aligned} X_s &= (x - x_s)cos(yaw_s) + (y - y_s)sin(yaw_s) \\ Y_s &= -(x - x_s)sin(yaw_s) + (y - y_s)cos(yaw_s) \\ \rho &= \sqrt{X_s^2 + Y_s^2} \\ \phi &= atan2(Y_s, X_s) \end{aligned}$$

- On calcule la probabilité d'occupation de la cellule en utilisant le modèle inverse de perception qui diffère selon la méthode utilisée.
- On se sert de cette probabilité ainsi que de la probabilité associée à la cellule de la vieille carte-sonar pour déduire la probabilité actuelle associée à cette même cellule (ici une formule de *Bayes* est utilisée ou une autre formule selon la méthode).

3.4 Simulation et Choix de la méthode de construction de carte

Avant de passer à la phase d'implémentation sur le FRMI, nous avons choisi de simuler le composant que nous comptons intégrer à la cartographie i.e. le module d'enrichissement de carte (voir figure 3.4 détaillant l'architecture). Cette simulation est réalisée en *Matlab* et elle vise essentiellement deux objectifs :

Algorithm 2 Boucle d'enrichissement

pour $i = 1...nbSonars$ **faire**
  Récupération de la mesure et la pose de chaque sonar
  $r = sonar(i).range$
  $x_s = sonar(i).pose.x$
  $y_s = sonar(i).pose.y$
  $yaw_s = sonar(i).pose.yaw$
  pour $i = i_{init}...i_{end}$ **faire**
    pour $j = j_{init}...j_{end}$ **faire**
      si $occup(i,j) \neq UNKNOWN$ et $occup(i,j) \neq OCCUPIED$ **alors**
        Calcul de la distance et l'angle de la cellule par rapport au sonar
        $[\rho, \phi] = positionCellule([i,j],[x_s, y_s, yaw_s])$
        Calcul du modèle inverse
        $p = modeleInverse(r, [\rho, \phi])$
        Mise à jour de la probabilité
        $occup(i,j) = miseAJour(p, occup(i,j))$
      fin si
    fin pour
  fin pour
fin pour

- Valider en simulation l'algorithme conçu pour le composant avec des données expérimentales récoltées sur le FRMI.
- Choisir la technique de calcul probabiliste la plus performante parmi trois techniques simulées.

Présentons tout d'abord les trois techniques de calcul à disposition.

3.4.1 Les troix méthodes

Pour le calcul probabiliste, nous étions confronté à faire un choix entre trois méthodes de calcul. Ces méthodes sont inspirées des articles de Ribo et Pinz (2001) et Gambino *et al.* (1996).

Ces deux articles ne parlent que de ces troix méthodes calculatoires et de leurs bases théoriques. Nous allons alors éviter d'exposer tous les détails théoriques et les formules mathématiques, quitte à se rendre à l'annexe A ou aux articles et à leurs références pour chercher ces détails, en résumant brièvement ici les différences majeures entre les différentes méthodes.

En effet, les trois méthodes diffèrent par le modèle inverse de perception utilisé.

Nous avons pour la première méthode [7] (« *Probabilistic approach* » : ProbA) un modèle simple

7. pour désigner les méthodes, nous allons adopter l'ordre dans lequel elle sont présentées dans l'article

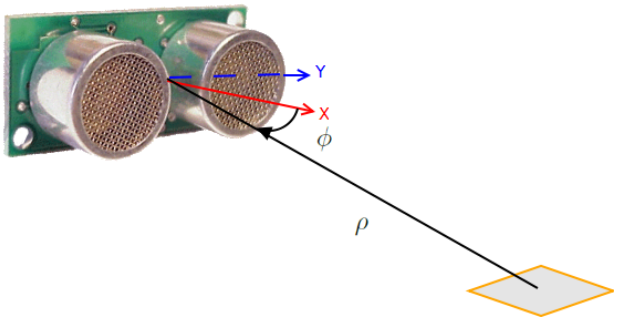

Figure 3.6 Figure montrant les coordonnées polaires d'une cellule dans le repère du sonar

et plutôt intuitif qui cherche à quantifier la probabilité d'occupation d'une cellule. Ce modèle est mathématiquement construit de façon à accorder une faible probabilité d'occupation aux cellules situées à l'intérieur du cône de perception, une probabilité élevée vers le bord de ce cône notamment autour de l'axe du capteur, et une probabilité neutre partout ailleurs. Bien évidemment, des constantes reliées au débattement angulaire du cône de détection, la portée et la précision du capteur sont à choisir avec soin pour approximer au mieux le comportement de nos sonars (Figure 3.7). Puis, c'est à partir de la propriété de *Bayes* que l'on a une loi de mise à jour.
Quant aux deuxième (« *Evidence theoretic approach* » : ETA) et troisième (« *Possibilistic approach* » : PossA) méthodes, elles utilisent toutes les deux un modèle inverse de perception double. Ce modèle permet de calculer deux probabilités, l'une quantifiant à quel point il est probable que la cellule soit libre, l'autre à quel point il est probable qu'elle soit occupée. Cette logique double ouvre la porte à la possibilité de considérer un cas où les mesures des sonars sont plus contradictoires (cellule libre et occupé en même temps) que tendant exclusivement à l'un des deux états (libre/occupée) (voir Figure 3.8). Ces deux dernières méthodes diffèrent alors essentiellement par leurs formules de mise à jour de la grille de cellules. Le calcul pour la méthode PossA se base sur l'accord d'un score à chaque cellule selon une logique de la théorie des ensembles. Tandis que la méthode ETA met à jour les grilles selon une formulation de la théorie de l'évidence de *Dempster-Schafer*.

3.4.2 Récolte et préparation des données expérimentales

Pour tester notre simulation, nous devons avoir des données expérimentales recueillies sur le FRMI. Pour ce faire, nous réalisons une expérience de cartographie en supposant que nous avons le composant d'enrichissement. On lance alors la cartographie par *GMapping*,

de Ribo et Pinz.

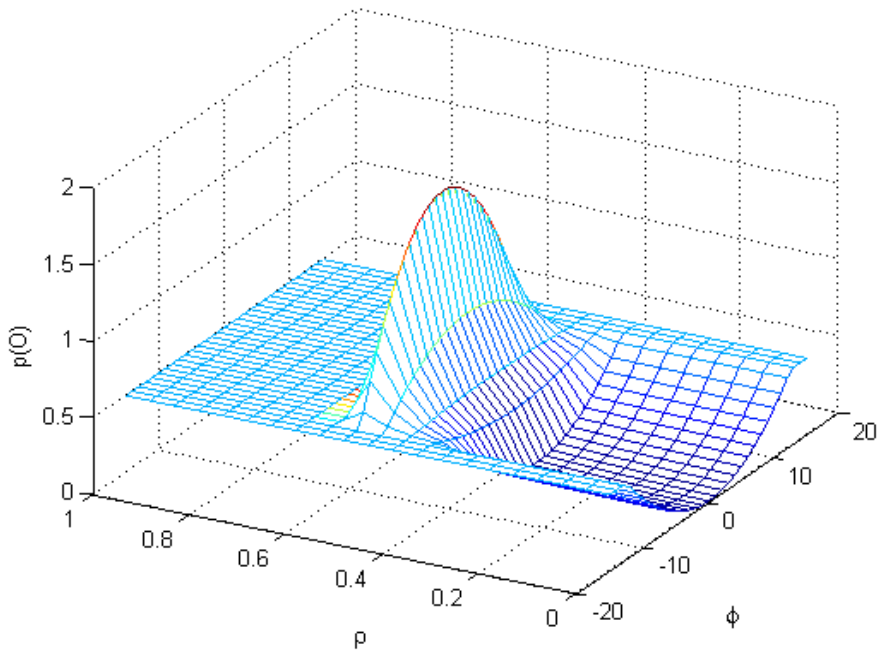

Figure 3.7 Modèle inverse de la première méthode pour une mesure de $0.5m$

on active les télémètres laser ainsi que les sonars. Et on navigue dans un milieu contenant des obstacles transparents tout en enregistrant les données nécessaires à notre simulation, à savoir les mesures des télémètres ultrasons, leurs poses dans le repère *Monde* et la carte-laser. Pour faciliter la récolte de données et surtout pour minimiser leur pré-traitement en Matlab, nous avons construit deux programmes de tâches bien précises :

- **sonarsPosPub** : Le système du FRMI étant implémenté sous *ROS*, nous profitons des fonctionnalités puissantes offertes par l'outil ***tf***[8] de *ROS* pour aller chercher intelligemment les transformations qui lient les repères des sonars à celui de la carte pour en déduire leurs poses dans le repère *Monde*. Ces poses sont simplifiées pour en extraire les valeurs de x_s, y_s et yaw_s avant de les publier.
- **sonarsRanPub** Ce module permet de recueillir les mesures de tous les sonars, publiées sur des canaux différents et de les réunir sur un seul canal de publication tout en faisant le tri des données valides (les mesures nulles sont éliminées).

Ces deux modules préparent également le terrain à l'accueil du composant à implémenter sur le fauteuil.

8. Pour plus d'informations sur cet outil puissant, vous pouvez consulter la page de sa documentation sur le wiki de *ROS*

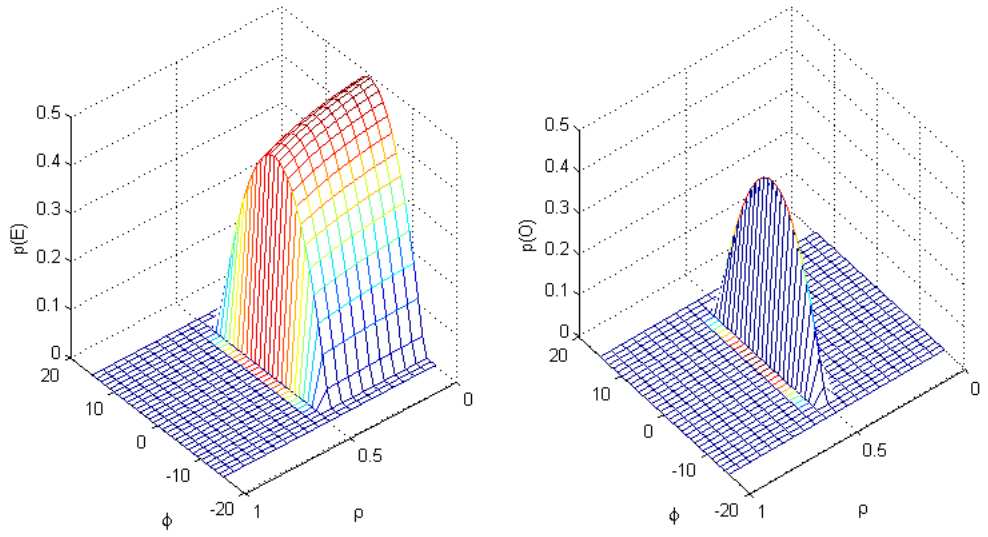

Figure 3.8 Modèle inverse de la deuxième et troixième méthodes pour une mesure de $0.5m$

Sous *Matlab* les traitements de préparation sont minimes. Les données enregistrées dans des *Bagfiles*[9] sont transformées en des structures de données classiques de *Matlab* en utilisant la bibliothèque *matlab_rosbag*. La carte-laser est ensuite reconstruite de façon à respecter les règles de parcours des tableaux sous *Matlab* et de façon à assurer l'adaptation correcte de l'information sur la position de l'origine de la carte.

3.4.3 Premiers résultats de simulation

On se propose dans un premier temps d'avoir un premier retour pour comparer qualitativement les méthodes de calcul probabiliste.

Une première expérience, où on place certains obstacles transparents dans l'entourage du FRMI, nous permet d'avoir un premier retour sur notre solution. Les différentes cartes obtenues sont données par la figure 3.9.

Ces premières cartes montrent une réussite assez satisfaisante à détecter et dessiner avec une certaine précision les obstacles transparents qu'il n'est pas très difficile de deviner leurs positions sur la carte-laser en regardant les cartes-sonar obtenues.
On remarque clairement l'efficacité de la technique à profiter, comme prévu, de la portée plus élevée des télémètres laser pour représenter les espaces explorés par les rayons laser au-delà des obstacles transparents.

9. Ce sont des fichiers d'enregistrement de données sous *ROS* grâce à l'outil *rosbag*. Consuler la documentation de rosbag pour plus d'informations.

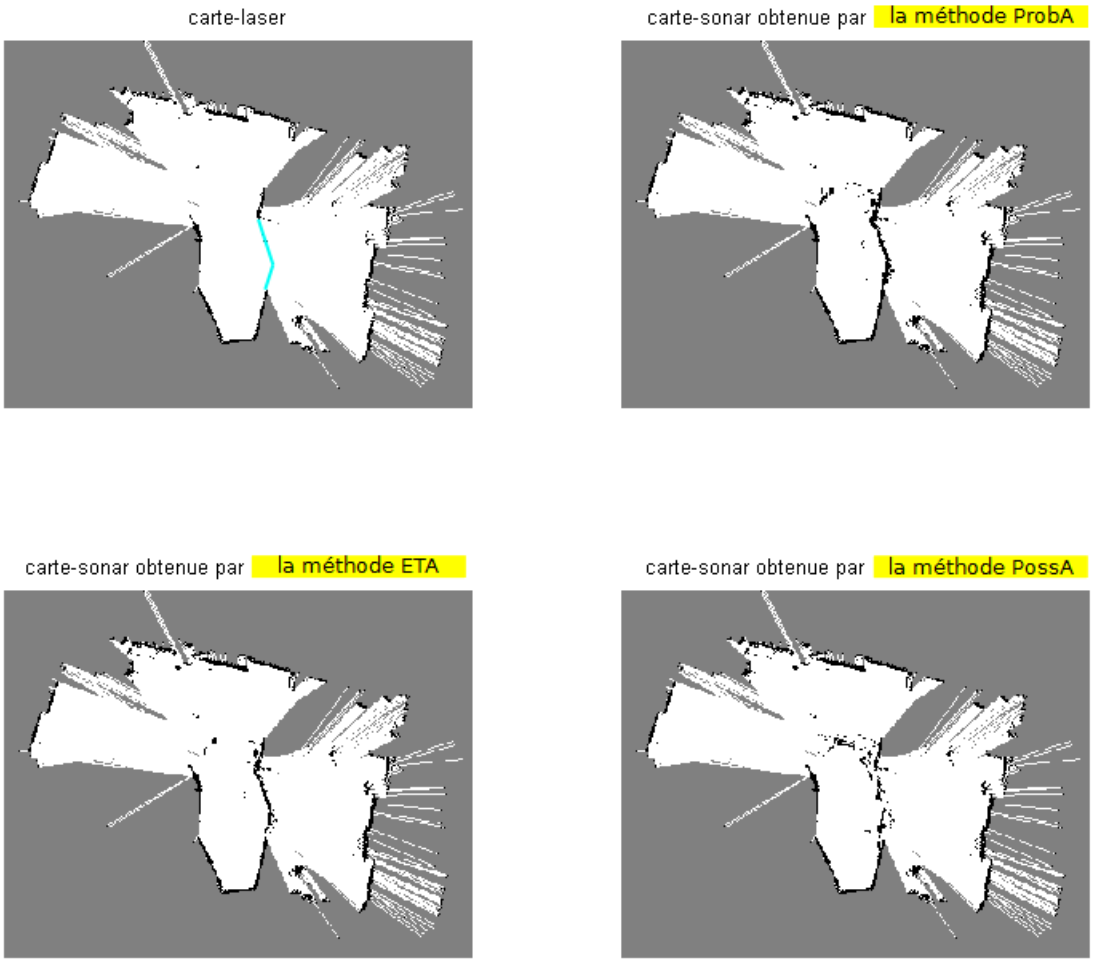

Figure 3.9 Premiers résultats de simulation du module d'enrichissement : carte-laser avec obstacles transparents rajoutés dessus en bleu (en haut à gauche) et les cartes enrichies pour les 3 méthodes

Quant à la comparaison des méthodes de calcul, on voit tout de suite la supériorité des méthodes ProbA et ETA par rapport à la méthode PossA. Cette dernière est nettement moins précise sur les obstacles et retourne un résultat plus bruité. Par contre, entre ProbA et ETA, on ne peut pas conclure immédiatement même si l'on pourrait être tenté de dire que la carte de la méthode ETA est plus propre. Mais, il faut remarquer, en contre-partie, que les obstacles transparents représentés sur la première carte-sonar sont plus consistants et « complets ».
Bien évidemment, il ne faut pas se précipiter à conclure sur le choix de la technique de calcul à retenir sans avoir recours à une méthode rigoureuse pour faire ce choix. Ceci est l'objet du paragraphe suivant.

Cette première expérience nous a permis de conclure sur la satisfaction du premier objectif de la simulation :
L'algorithme conçu pour le composant d'enrichissement de carte est fonctionnel et retourne des résultats « corrects » avec des données expérimentales récoltées sur le FRMI.

3.4.4 Comparaison expérimentale des méthodes

Nous cherchons à effectuer une comparaison rigoureuse entre les méthodes pour faire un choix justifié de la meilleure pour notre module. Nous réalisons alors deux expériences de cartographie en milieu contrôlé.
D'abord, on fait une première cartographie d'un milieu qu'on prépare préalablement en plaçant des obstacles à des endroit précis. Les obstacles sont tous opaques pour une première cartographie en utilisant bien sûr le module que nous avons, i.e. le *GMapping*, avec les seules mesures retournées par les lasers. Ensuite, on remplace deux obstacles par deux autres de mêmes tailles mais qui sont cette fois-ci transparents et on recommence la cartographie de la même pose de départ et en essayant de parcourir le même trajet. Au cours de cette dernière expérience, on fait le collecte des données de la façon présenté plus haut pour la simulation. Nos cartes-laser obtenues à l'issue de ces deux expériences sont présentée à la figure 3.10. Nous avons également représenté la moyenne des deux cartes pour vérifier la superposition.

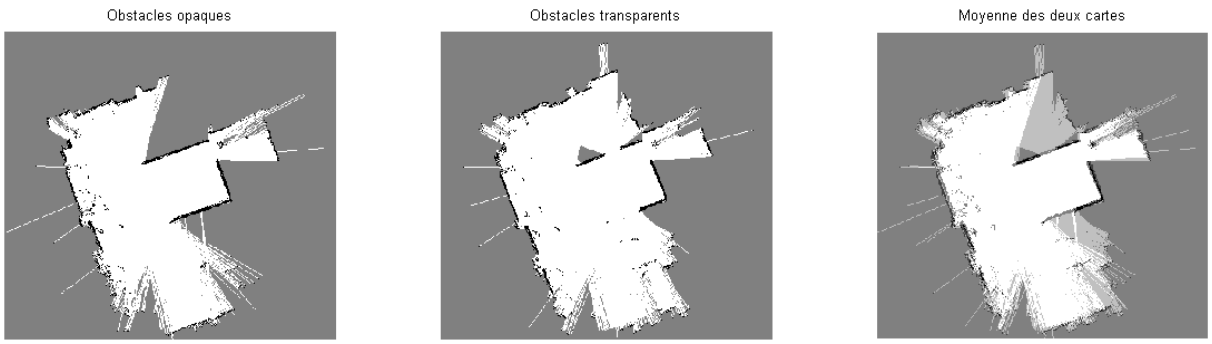

Figure 3.10 Cartes-laser obtenues dans un milieu contrôlé sans et avec obstacles transparents

Dans une étape suivante, nous avons effectué la simulation des trois méthodes en utilisant les données collectées à la seconde cartographie (voir Figure 3.11 où la première carte est la carte-laser crée en milieu opaque).

Maintenant, on a tout ce qu'il faut pour calculer des scores quantifiant la précision de l'enrichissement.
On examine, pour les calculer, la différence GC_d entre les deux grilles de cellules GC_i et

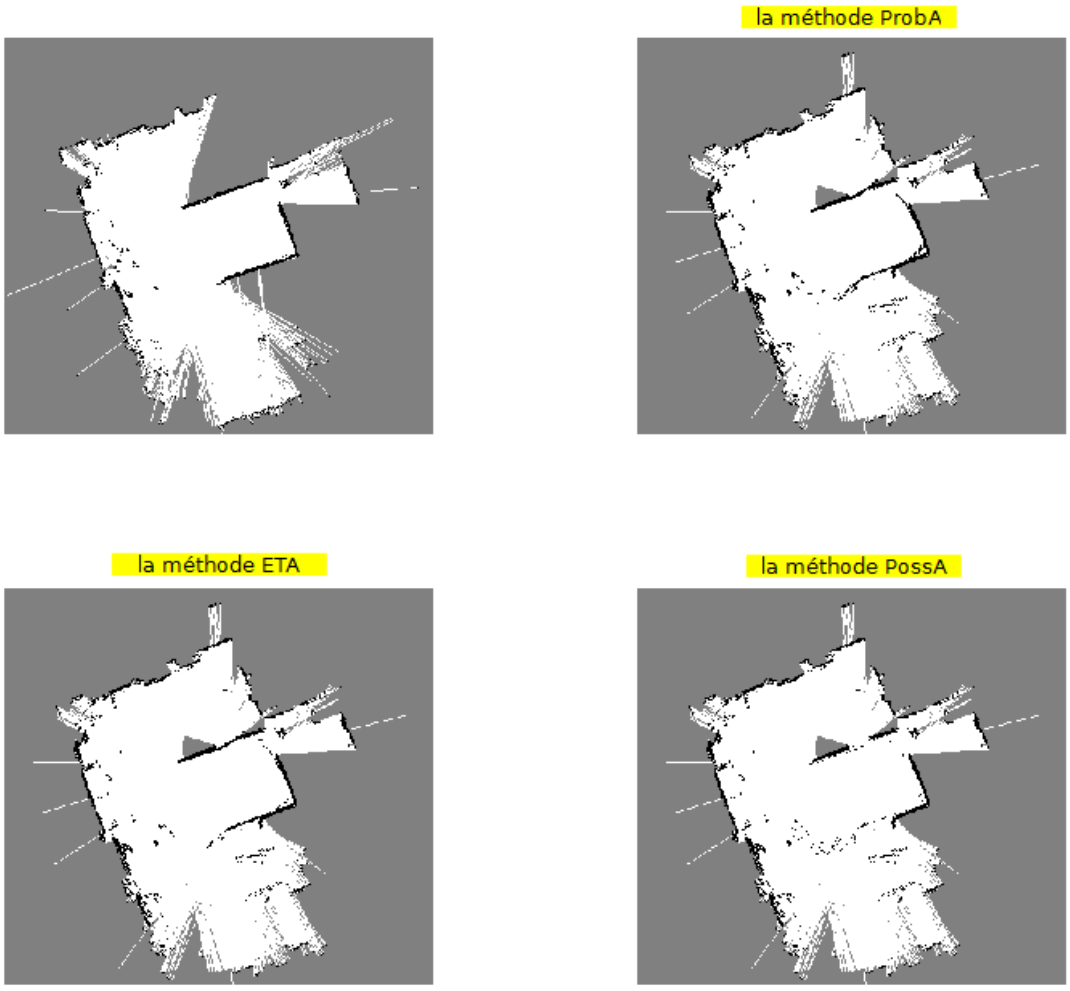

Figure 3.11 Résultats de simulation pour les troix méthodes de calcul probabiliste

GC_{lt} où GC_i est celle de la carte de la méthode $i \in \{1, 2, 3\}$ et GC_{lt} est celle de la carte-laser obtenue en milieu contenant des obstacles transparents. Si la valeur du pixel (i, j) est non nulle i.e. $GC_d(i, j) \neq 0$, alors on accorde selon la comparaison de $GC_i(i, j)$ et $GC_{lo}(i, j)$ (grille de la carte-laser crée en milieu opaque) un point de bonus ou une pénalité à la méthode correspondante.
On associe à chaque méthode le même score initial :

$$\begin{aligned} s_1 &= 200 \\ s_2 &= 200 \\ s_3 &= 200 \end{aligned}$$

On accorde un point de bonus à chaque pixel faisant partie d'un obstacle dessiné correcte-

ment sur la carte-sonar.

Soit p_n et p_d les pénalités associées respectivement aux éléments non dangereux et aux éléments dangereux.

- Un élément est dangereux si un élément d'un obstacle n'est pas représenté en tant que tel sur la carte-sonar.
- Un élément est non dangereux si on a un « pixel obstacle » là où l'espace est libre.

On choisit alors $p_d \leq p_n \leq 0$ car il est logique de pénaliser les éléments dangereux plus que les éléments non-dangereux. Si on fixe p_d à $p_d = -0.5$. Les scores finaux varient alors avec le rapport :

$$f = p_n/p_d$$

En faisant varier f entre 0 et 1, on obtient alors les courbes de scores en fonction du facteur f. Ces courbes sont données à la figure 3.12.

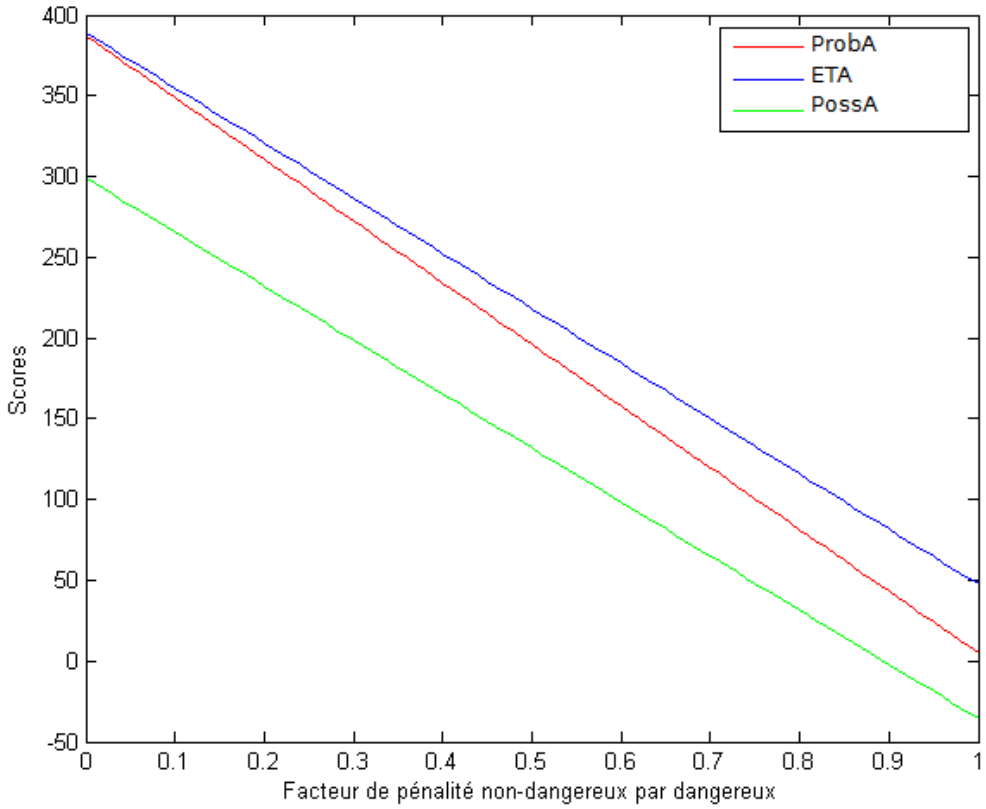

Figure 3.12 Scores des différentes méthodes de calcul probabiliste en simulation avec des données récoltées sur le FRMI en déplacement dans un milieu contrôlé

La courbe bleue de la méthode ETA reste au-dessus des courbes de scores des deux autres

méthodes. Ainsi, le score de cette méthode est toujours le meilleur quelque soit la valeur du facteur f. On voit également que les deux courbes des méthodes ProbA et ETA reflètent des scores relativement rapprochés et clairement supérieurs aux scores de la PossA, ce qui rejoint nos observations qualitatives au paragraphe 3.4.3.

Nous concluons alors que la méthode à retenir pour l'implémentation sur le FRMI du composant d'enrichissement de cartes par les mesures sonar est la ETA.

Dans les deux dernières parties, nous allons exposer brièvement quelques détails importants reliés à l'implémentation pour mettre le point sur certaines difficultés pratiques rencontrées à cette phase de la réalisation (voir 3.5). Et puis, nous allons conclure dans 3.6 par la présentation de résultats de test de la cartographie optimisée sur le FRMI.

3.5 Éléments importants de l'implémentation

3.5.1 Lecture et fréquences de publications

Comme nous l'avons spécifié (au 3.4.2, paragraphe **Récolte et préparation des données expérimentales**), les mesures des sonars sont publiées sur un seul canal en éliminant les données non valides (valeurs nulles ou absence de mesure par exemple). Si nous éliminons des mesures, il faut en contrepartie être en mesure de reconnaitre le sonar (parmi 6 sonars) à l'origine de chaque détection acceptée parce que sinon on ne peut pas connaitre la pose correspondante à chaque sonar.
La solution consiste à classer les poses des 6 sonars dans un ordre spécifique (de droite à gauche) et associer le numéro d'ordre convenable à chaque mesure retenue dans les messages contenant les mesures.

D'autre part, en système réel, les détections et traitements d'enrichissements se font en temps réel. Pas de traitement, à un instant donné t, sans des mesures assez récentes retournées par les sonars mais aussi pas de traitement sans avoir une connaissance assez précise des poses des sonars dans le repère *Monde*, une connaissance qui ne soit pas très en retard par rapport à la fréquence de détection.
Les modules **sonarsPosPub** et **sonarsRanPub** doivent ainsi publier au même rythme. Nous choisissons alors une même fréquence [10] de publication aux deux modules. De plus, nous veillons à bloquer le traitement tant que nous n'avons pas les deux informations indispensables à celui-ci à chaque période du module d'enrichissement [11], à savoir les mesures des distances aux obstacles et les poses des sonars.

10. nous avons pris une période de 0.15sec
11. cette période est choisie égale à 0.2sec

3.5.2 Formats des cartes

Les cartes reçues par le module (ou cartes-laser comme nous les appelons dans ce rapport) se présentent sous le format de grilles d'occupation de *ROS*[12]. La grille contenant l'information relative à la carte est alors un tableau d'entiers de dimension un, qui est la transformation en « Row-Major Order » de la véritable grille de cellules.
À cause de l'aspect double du modèle de perception de la méthode de calcul choisie, les grilles des cartes-sonar ne peuvent être représentées exactement de la même façon. C'est ainsi que nous avons créé un type de message spécial, que *ROS* ne connait pas, inspiré du format déjà existant. Ce nouveau format garde l'avantage de transport des informations de position de l'origine dans le repère *Monde* ; avantage dont nous avons besoin pour ne pas perdre cette donnée à la « re-conversion » au format classique pour la publication de la carte enrichie. Mais, la représentation de l'information relative à la carte change en adoptant deux tableaux de *Doubles* représentant en « Row-Major Order » la double grille de cellules imposée par le modèle inverse de perception de la méthode choisie (voir 3.4.2).

Pour publier cette carte de format différent dans le réseau du système (pour qu'elle puisse être accessible par d'autres modules), il faut alors effectuer un post-traitement spécifique qui permet une conversion correcte au format classique.
La carte-sonar est convertie selon les règles suivantes :

- Associer les bonnes valeurs spécifiques aux cellules appartenant aux obstacles détectés par les télémètres laser et aux cellules d'occupation inconnue, qui sont déjà marquées à la phase d'enrichissement.
- Pour les autres cellules, on calcule la différence entre les valeurs d'occupation et de non-occupation associées à chaque cellule. Si la différence est supérieure à un seuil choisi empiriquement, elle est alors occupée, sinon, elle est libre.
  Le seuil est une valeur qui dépend de beaucoup de facteurs : le nombre et la disposition des sonars, leurs performances, la vitesse de navigation lors de la cartographie etc...

3.6 Résultats de test en système réel

3.6.1 Test en laboratoire

On utilise des obstacles opaques et transparents à notre disposition au laboratoire pour mettre une scène en place. On y teste alors notre programme implémenté sur le FRMI. Le résultat (Figures 3.13, 3.14, 3.15) montre une très bonne efficacité d'enrichissement de la carte-laser. La représentation des obstacles qui ne sont pas vus par les télémètres laser ainsi

12. messages de type **nav_msgs/OccupancyGrid**

que leurs positions sont correctes. On remarque même qu'on est parvenu à avoir un résultat où le bruit est très faible.

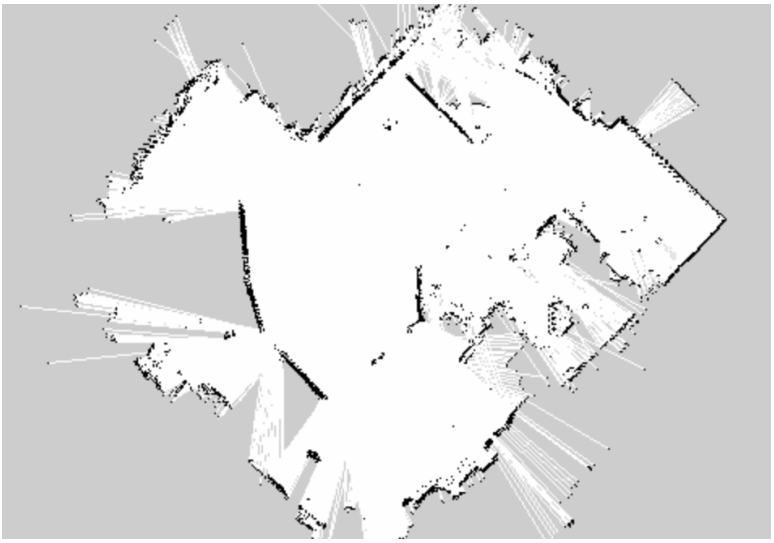

Figure 3.13 Résultat de test en labo : carte-laser

Figure 3.14 Carte-laser où les obstacles transparents sont indiqués

3.6.2 Test en milieu public

A l'étage 5 du pavillon *Lassonde* de l'École Polytechnique, nous effectuons la cartographie en passant par des endroits représentants des défis à notre module d'enrichissement.
Les tests ont été effectués à l'heure du lunch afin de profiter de la présence de nombreux

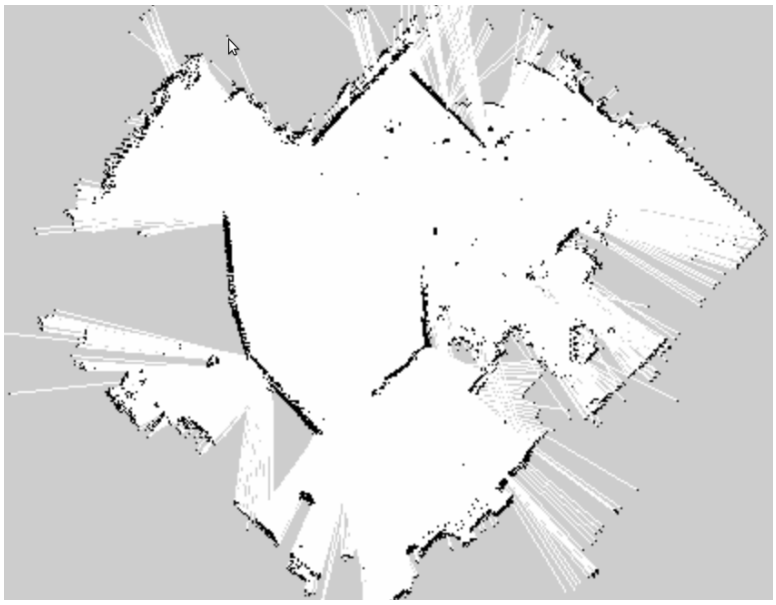

Figure 3.15 Résultat de test en labo : carte-sonar

passants pour maximiser la difficulté.
Le FRMI rencontre des obstacles transparents à 3 emplacements en effectuant la navigation dans l'étage :

- la fenêtre qui donne sur l'extérieur à côté des ascenseurs
- la façade de la salle de réunions
- la façade du laboratoire d'enseignement « Analyse et traitement des signaux »

Les cartes obtenues sont données par les figures 3.16 et 3.17 où nous avons montré les emplacements des obstacles en vitre.

Elles confirment bien l'efficacité de cette technique et montrent que l'integration du composant d'enrichissement permet de combler les limites d'une cartographie basée uniquement sur le *GMapping* et les télémètres laser.

3.6.3 Discussion

Le module d'enrichissement de carte par données sonars retourne bien des cartes enrichies efficacement. Cependant, on pense que les résultats d'enrichissement sont étroitement liés à divers facteurs :

- Qualité des télémètres ultrason utilisés. Nous pensons que nos mesures de sonars étaient relativement médiocres. En effet, trois de nos six sonars seulement retournaient des données acceptables à chaque prise de mesure de façon régulière.
- Portée des télémètres ultrason. Nos sonars ont dans la pratique une portée inférieure à

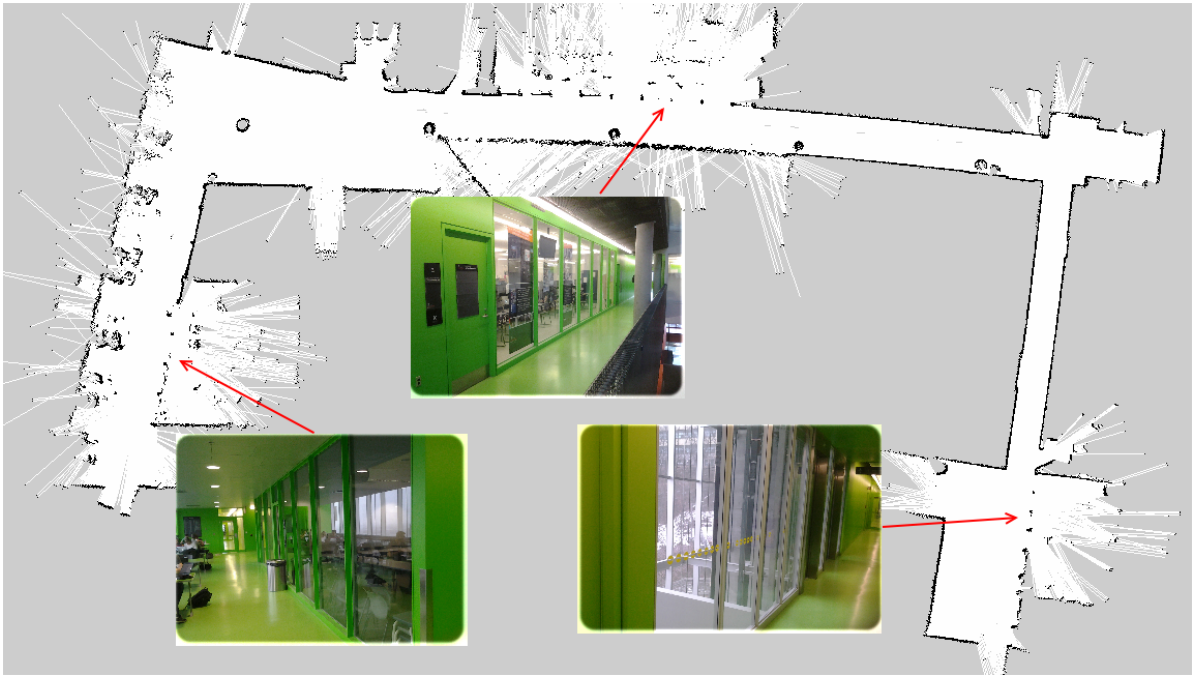

Figure 3.16 Résultat de test à l'étage 5 du pavillon Lassonde : carte-laser

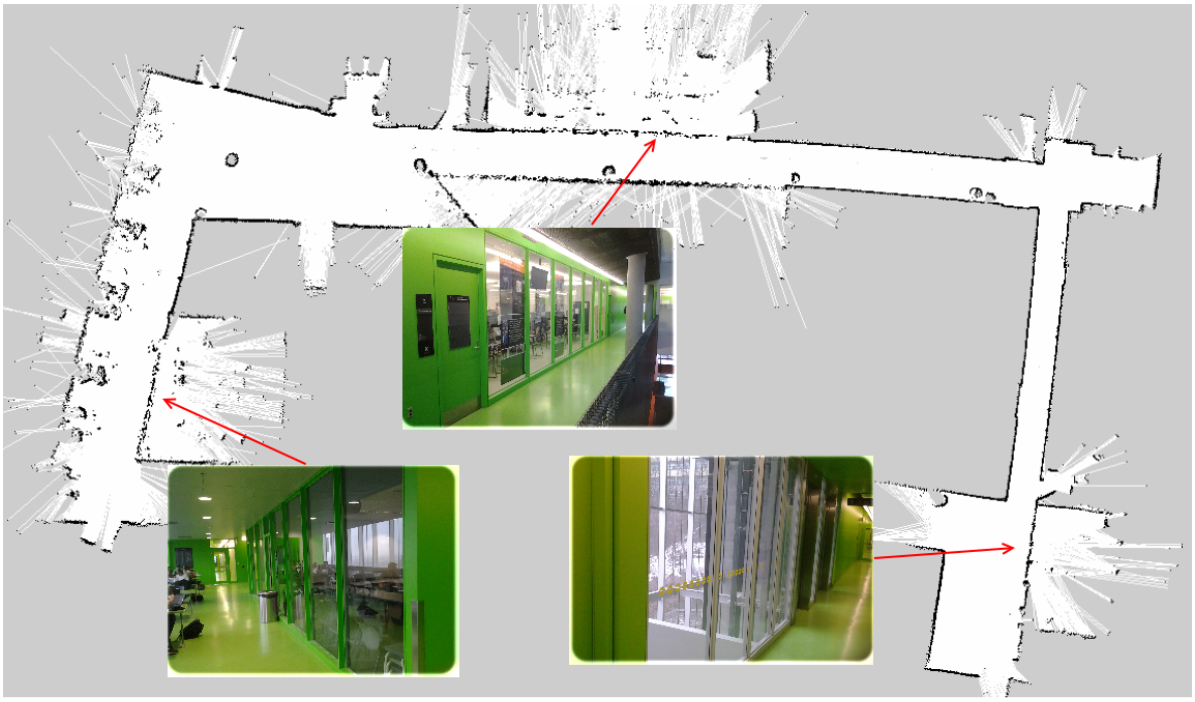

Figure 3.17 Résultat de test à l'étage 5 du pavillon Lassonde : carte-sonar

$1m$. Selon les spécifications, la portée devrait être de $1.5m$ mais nous avons remarqué que les mesures ne sont pas fiables au-delà de $1m$. cela est peut-être du à l'emplacement

proche du sol des sonars sur le FRMI ce qui fait que l'onde sonore touche le sol en se propageant plus loin.

– Nombre de sonars. Avec 6 sonars, il faut rouler à une vitesse assez faible (inférieure à $0.6m/s$) à proximité des obstacles pour donner le temps au sonars de retourner suffisamment de données pour l'enrichissement. Sinon, il faut faire plusieurs passages pour assurer ceci.

Les télémètres ultrason sont capables de détecter les obstacles opaques aussi et pas seulement les obstacles transparents. Il est alors légitime de se poser la question : notre composant peut-il compléter un obstacle opaque manquant dans une carte donnée ?
Nous avons bien testé cette idée et nous présenterons au cours du chapitre suivant des exemples de cette expérience.

CHAPITRE 4

CONSTRUCTION AUTOMATIQUE DE CARTES EN GRILLES D'OCCUPATION A PARTIR DE PLANS D'ARCHITECTURE

Les méthodes de cartographie par SLAM comme *Le GMapping* nécessitent une exploration « coûteuse » de l'environnement à cartographier. Il serait alors intéressant de trouver un moyen qui nous permet une construction sans exploration.

Dans ce chapitre, nous allons présenter une deuxième technique d'optimisation de la navigation du FRMI : la construction automatique d'une carte, se présentant sous le format d'une grille d'occupation (le format utilisé par le système de navigation du FRMI), à partir d'un plan d'architecture (ou carte CAO).
Notre étude de la revue de littérature relative à la cartographie robotique (chapitre 2), nous incite à penser qu'il s'agit d'une solution de construction toute nouvelle (même si l'idée d'exploiter les cartes d'architecture dans le domaine de la navigation a été traitée par certains articles récents comme les travaux de Schafer *et al.* (2011)). En effet, nous pensons que notre solution est la première à résoudre le problème de construction d'une grille d'occupation à partir d'un plan de sol.
Nous allons d'abord présenter une introduction au format des cartes d'architecture intéressant pour notre recherche 4.1. Ensuite, nous détaillerons l'algorithme utilisé pour la construction 4.2. Enfin, l'implémentation du programme sur le fauteuil et son application à des cartes d'étages de polytechnique sont exposées dans une troisième partie 4.3 ainsi que différents exemples de l'utilisation d'une de ces cartes générées dans une quatrième partie 4.4 avant de conclure à la dernière 4.5.

4.1 Plans d'architecture et formats

4.1.1 Notions générales

Un plan (ou dessin d'architecture[1]) est une représentation selon des règles techniques spécifiques d'un édifice donné. Plusieurs vues sont connues en dessin architectural. Celle qui nous intéresse ici est le **plan de sol** (vue principale en architecture).
« Le plan de sol est le principal dessin d'architecture. C'est une vue de dessus qui représente la disposition des espaces dans un bâtiment, à la manière d'une carte, pour un étage du

1. Cette introdution de la notion de dessin d'architecture est inspirée de *Wikipédia*

bâtiment. Techniquement, c'est une section horizontale d'un bâtiment (conventionnellement à un mètre au-dessus du sol), représentant notamment les murs, les portes et les fenêtres. »[2].

De nos jours, les plans d'architecture sont de plus en plus réalisés en exploitant les outils de conception assistée par ordinateur (CAO). Parmi les outils les plus utilisés aujourd'hui, le plus utilisé peut-être, le logiciel *AutoCAD* créé en décembre 1982 par *Autodesk*.
Ce logiciel permet de créer des fichier CAO de format *DWG*. Et afin de faciliter l'échange des fichiers CAO entre les différents outils de conception, *Autodesk* a créé le format *DXF*. Ce format est ainsi supporté par presque tous les outils de conception et la conversion de et à ce format est généralement possible pour plusieurs autres formats de fichiers CAO d'où son nom « format d'échange de dessin ».

4.1.2 Structure du format *DXF*

Puisque presque tous les formats de fichiers CAO sont convertissables au format *DXF*, il est alors possible de générer sans problème le fichier *DXF* correspondant à un plan donné. C'est en partie pour cette raison que nous basons notre solution de génération de carte à partir de cartes CAO sur ce format en particulier c'est-à-dire que le fichier CAO du plan servant d'entrée à notre processus de génération doit être disponible dans ce format.
L'autre raison justifiant ce choix réside dans la disponibilité en libre consultation de la structure de ce format. En effet, le document technique intitulé « DXF Reference » Autodesk (2010) publié régulièrement par *Autodesk* résume les éléments et les régles de la structure d'un fichier DXF et donne même des exemples simples de méthodes de lecture et d'écriture de ces fichiers pour programmer une interface (voir pages 227-234 de Autodesk (2010)).

Introduisons ici brièvement les éléments de cette structure qui sont importants pour notre travail.
L'élément constitutif de base d'un fichier *DXF* est la paire code/valeur :

- **le code** est un chiffre unique représentant une entité, grandeur, etc. spécifique. C'est en quelque sorte l'identité de l'élément qu'on quantifie, mesure ou spécifie par une valeur donnée. Par exemple, le code '8' est associé au nom du layer (calque) d'une entité donnée, le code '10' désigne l'abscisse du premier point (comme l'extrémité d'un trait, centre d'un arc) d'une entité donnée.
- **la valeur** est la quantité, le nom, l'option, etc. associée au code.

Ainsi, c'est avec des paires code/valeur bien ordonnées selon une structure bien déterminée que l'on construit un fichier *DXF* correct et lisible et donc un graphique.
Dans la figure 4.1, nous montrons cette structure tout en mettant en valeur les sections et les parties qui nous intéressent le plus. En effet, la section *HEADER* (en-tête) contient les

2. Voir paragraphe **plan de sol** de *Wikipédia*

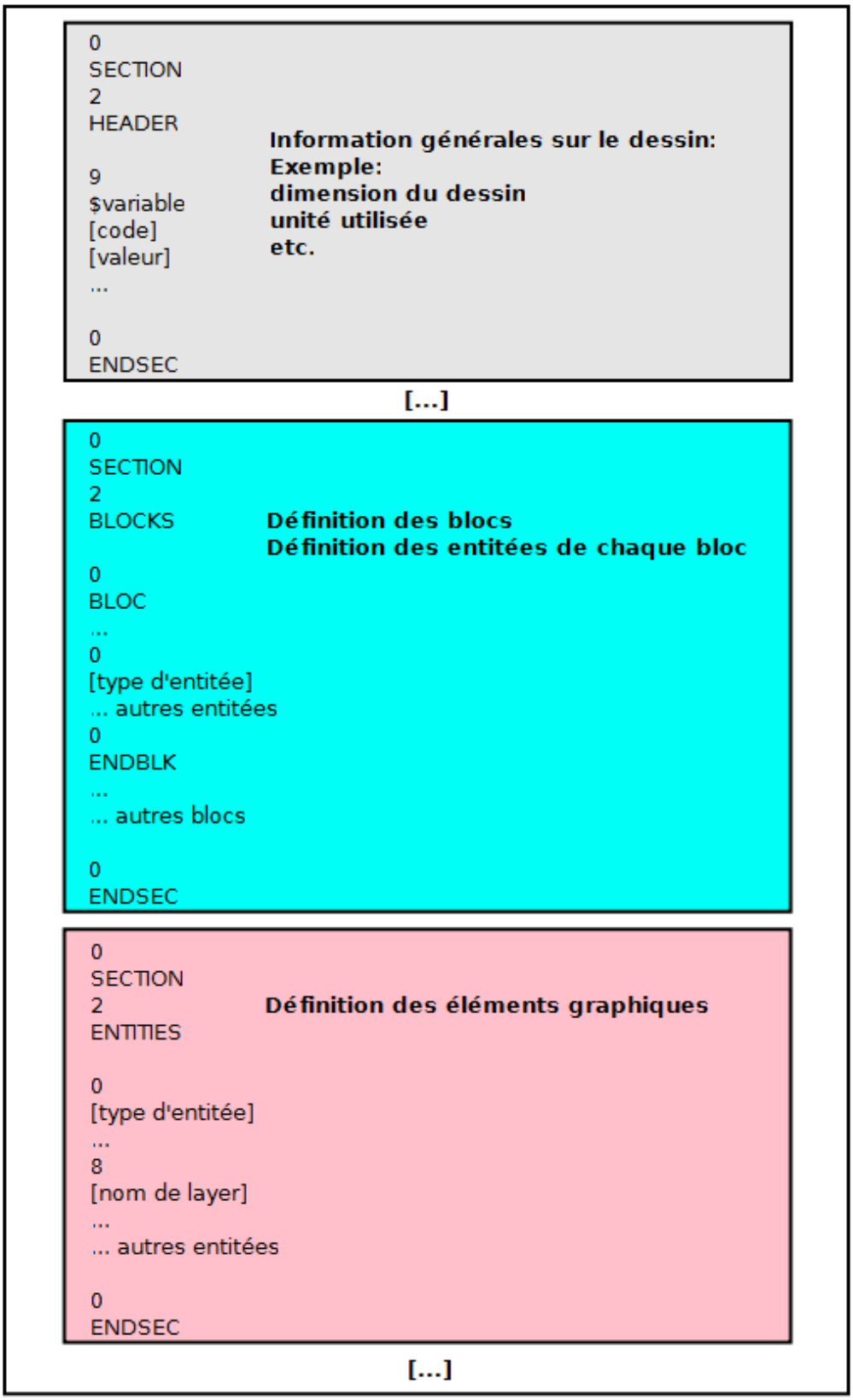

Figure 4.1 Structure générale d'un fichier DXF : Mise en valeur des éléments importants

informations générales du graphique. C'est ici que nous sommes sensés aller chercher des informations indispensables à la construction de notre carte tels que la largeur et la hauteur convenables ainsi que l'unité dans laquelle les longueurs sont exprimées.
Quant aux sections *BLOCKS* et *ENTITIES*, elles regroupent tous les éléments graphiques que nous avons besoin de reconstruire sur notre carte en grille de cellules. Dans un fichier *DXF* représentant un plan architectural de sol, nous cherchons tous les éléments suivants :

- **Lignes** : servent en général à représenter des murs, des fenêtres, des escaliers, des ascenceurs, etc.
- **Polylignes** : servent en général à la représentation des pièces, piliers, etc.
- **Cercles** : servent à la représentation de piliers cylindriques en général.
- **Arcs** : servent à dessiner les ouvertures de portes, les rebord d'escaliers, etc...

Bien évidemment, une carte CAO contient généralement des éléments graphiques répartis sur différents calques (*layers*) dont certains représentent des entités purement fictives tels que les axes et les cotes. Pour éviter alors de dessiner des obstacles imaginaires en prenant certaines entités pour des éléments constitutifs du bâtiment ou de l'étage à cartographier, notre solution prévoit une première phase de traitement où l'on liste tous les layers et on les met à disposition de l'utilisateur pour choisir ceux qui sont les plus convenables.

4.2 Description de la solution

Précédemment, nous avons introduit les notions de base nécessaires à la compréhension de notre solution. Maintenant que les bases se sont clarifiées, nous allons parler en détail de la solution.

4.2.1 Entrées et Sorties

Les Entrées

Les Inputs du module sont :

- **Le plan architectural :** Il s'agit d'une carte CAO qui est enregistrée dans un fichier sous le format *DXF*
- **La résolution de construction :** C'est la taille en mètres de la cellule de la grille d'occupation à construire.
- **La liste des calques retenus :** L'utilisateur choisit une liste de calques parmi ceux formant la carte et les donne en entrée au module (on voit le module comme un bloc unique pour le moment).

Les Sorties

- **La carte générée :** Il s'agit de la carte que nous voulons construire et qui est supportée par le système de navigation du FRMI.
- **Le plan simplifié :** Il s'agit d'un fichier *DXF* simplifié écrit par notre module afin de réduire le plan de sol original en éliminant toutes les informations inutiles pour la génération de la carte de navigation. En particulier, ce plan élimine entre autres toutes les entités qui n'appartiennent pas aux calques qui ne figurent pas dans la liste de calques choisis par l'utilisateur. Ce plan élimine également toute entité ne pouvant être un élément constitutif d'un édifice.
- **La liste de tous les calques du plan original :** On fournit à l'utilisateur une liste de tous les calques contenant des lignes, polylignes, cercles et arcs pour lui permettre d'en choisir les plus appropriés

4.2.2 Structure de l'algorithme

Le module a deux composantes principales associées en série (voir Figure 4.2) :

- une composante permettant de préparer une carte simplifiée à partir de la carte CAO originale en profitant des choix de calques de l'utilisateur. Ce composant permet également d'extraire certaines informations indispensables à la construction de la grille d'occupation. Ainsi, nous nommons ce composant : « **Le composant de pré-traitement** ».
- une composante qui reçoit les données de sortie du pré-traitement pour construire la carte en grille d'occupation. Ce composant est appelé : « **Le composant de construction** ».

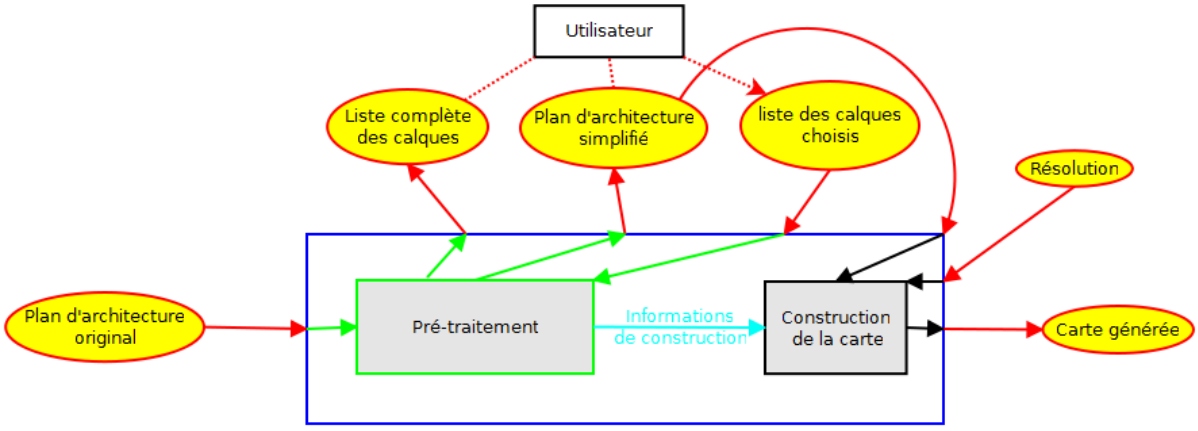

Figure 4.2 Structure du module et flux de données

Sur la Figure 4.2, on a mis en valeur, en plus des composants et des Inputs et Outputs, l'interaction informationnelle entre le module et l'utilisateur. Plus précisément, on montre ici le rôle de l'utilisateur qui se limite à la phase de pré-traitement.
En effet, l'utilisateur consulte la liste complète des calques et il se charge de la tâche de choisir une compilation de calques parmi cette liste. Le composant de pré-traitement génère, en fonction, un plan simplifié que l'utilisateur peut visualiser au moyen d'un outil de CAO pour, éventuellement, modifier en fonction son choix de calques.

Le paragraphe suivant aide à mieux comprendre cette interaction entre utilisateur et composant de pré-traitement en le décortiquant.

4.2.3 Pré-traitement du plan

Le pré-traitement se fait en trois temps.

Liste complète des calques

D'abord, le fichier du plan original est lu et exploré pour en extraire tous les noms de calques. Cette étape du traitement ne s'effectue qu'une seule fois par fichier.
Pour réaliser ce travail, nous faisons appel à la fonction de lecture *ReadDXF* proposée à la page 229 du Autodesk (2010) pour extraire les valeurs associées au code '8' des entités graphiques intéressantes pour notre module, à savoir les lignes, les polylignes, les cercles et les arcs. Ces entités sont cherchées dans les sections *BLOCKS* et *ENTITIES* du fichier *DXF* de la carte CAO originale.

Choix des calques de construction

Dans un deuxième temps, notre programme n'intervient pas et c'est le rôle de l'utilisateur de supprimer de la liste des calques retournée à la première phase tous les calques dont il veut enlever les entités.

Écriture du plan simplifié

Enfin, le composant crée à partir de cette liste de calques réduite un fichier *DXF* simplifié. La fonction permettant d'effectuer ce travail de reconstruction est inspirée de la fonction d'écriture *ReadDXFPolygon* de la page 233 du Autodesk (2010). Notre fonction parcourt tout le fichier *DXF* du plan original à la recherche d'éléments graphiques utiles (i.e. lignes, polylignes, cercles ou arcs) tout en écrivant un fichier de structure minimale. À chaque fois qu'un élément est trouvé, son nom de calque est examiné pour vérifier s'il figure bien dans la liste fournie par l'utilisateur. Dans ce cas, il est intégré dans le fichier simplifié sous un nom

de calque unique et commun à toutes les entités. En d'autres termes, on redessine tous les éléments graphiques intéressants sur le même calque.

Rôle de l'utilisateur

Ces deux dernières étapes du pré-traitement peuvent être reproduites en rectifiant à chaque fois la liste de calques en fonction du résultat obtenu jusqu'à ce que l'utilisateur soit satisfait du plan CAO simplifié. Bien évidemment, on visualise le fichier *DXF* retourné par le composant de pré-traitement au moyen d'un des nombreux outils de visualisation (dont beaucoup sont disponibles gratuitement sur la toile) ou de conception.
Mais, comment peut-on savoir si la carte simplifiée que nous avons est satisfaisante ? Notre objectif est de construire une carte utile pour la navigation globale du FRMI. Donc, il est évident que nous voulons garder le maximum d'obstacles sur la carte simplifiée tout en éliminant le maximum de détails superflus et si possible tout « artéfact » ne représentant pas une partie de l'édifice « réel » comme les écritures explicatives, les axes et les cotes qu'on rencontre communément sur les plans architecturaux. Pour rencontrer cette objectif, on choisit les calques à garder en sachant que leurs noms sont d'habitude révélateurs de la nature des éléments représentés et que la séparation entre les éléments de construction et les autres détails est généralement la pratique répandue dans la réalisation des plans par leurs dessinateurs.

Informations de construction

A la figure 4.3, nous résumons la structure du composant de pré-traitement.
Reste un dernier point à clarifier : le pré-traitement retourne des informations nécessaires à la construction de la grille d'occupation. De quelles informations avons-nous besoin et comment les extraire ?
Les informations nécessaires qu'il faut extraire du plan original sont :

- l'unité dans laquelle sont exprimés les longueurs dans le fichier *DXF*
- Les coordonnées du coin bas gauche du plan
- Les coordonnées du coin haut droit du plan

Le besoin de connaitre l'unité des longueurs exprimées dans le fichier est évident. Quant aux coins extrêmes du plan, ils servent, associés à la résolution, à calculer les dimensions de la grille de cellules à initialiser. Et ils permettent également d'avoir une référence pour pouvoir localiser les éléments graphiques au moyen de leurs données fournies dans le plan architectural.
Pour trouver et prélever ces informations du fichier *DXF*, il suffit de chercher dans le *HEA-*

DER les variables *$INSUNITS*, *$EXTMIN* et *$EXTMAX*. La valeur associée au code '70' de *$INSUNITS* donne le choix de l'unité. Les valeurs associées aux codes '10' et '20' de *$EXTMIN* et *$EXTMAX* permettent de déterminer les coordonnées des coins extrême du graphique.

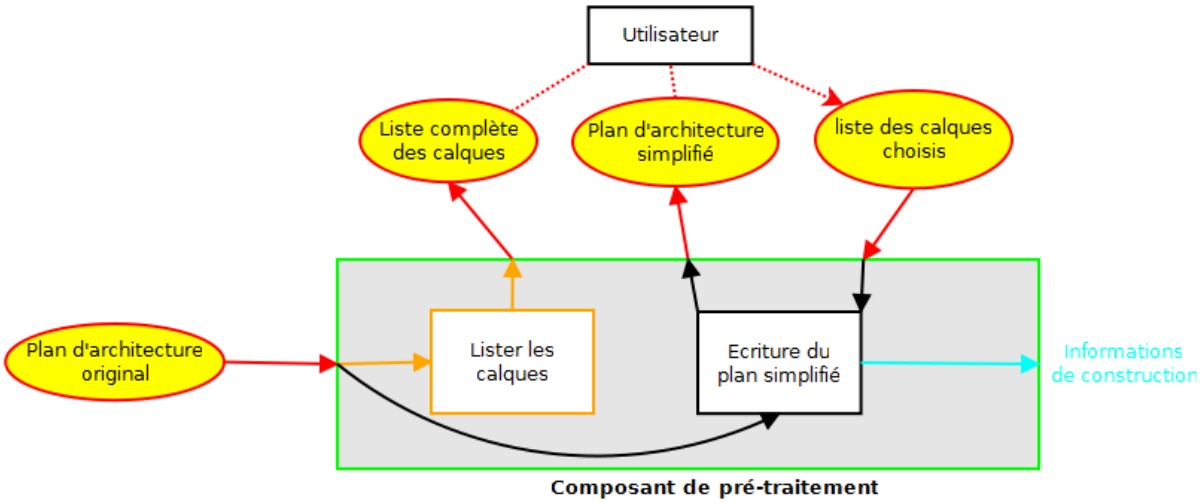

Figure 4.3 Structure du composant de pré-traitement

4.2.4 Construction de la carte

La phase de construction de la carte est celle qui va nous permettre de convertir une carte codée dans un fichier *DXF* en carte sous format de grille d'occupation.
La fonction de ce composant est alors d'extraire toutes les lignes, les polylignes, les cercles et les arcs du plan simplifié issu de la phase de pré-traitement, et de les « dessiner » sur une grille de cellules. Ceci est effectué en 3 étapes.

Extraction des éléments graphiques

Comme nous les avons cités maintes fois dans ce qui précède, les éléments graphiques recherchés sont les lignes, les polylignes, les cercles et les arcs. Le tableau 4.1, rassemble les données permettant de les trouver et de les définir géométriquement.

Nous créons alors pour chaque type d'objet une structure de données adaptée et on liste les éléments de façon à avoir 3 listes d'éléments :

- Une liste des lignes à dessiner
- Une liste des cercles à dessiner

Tableau 4.1 Éléments graphiques, leurs noms d'objet et leurs codes utiles

Élément	Nom de l'objet	Codes recherchés
ligne	LINE	(10, 20) : coordonnées de la première extrémité (11, 21) : coordonnées de la seconde extrémité
polyligne	LWPOLYLINE	90 : nombre de sommets (10, 20) : coordonnées d'un sommet
cercle	CIRCLE	(10,20) : coordonnées du centre 40 : rayon
arc	ARC	(10, 20) : coordonnées du centre 40 : rayon (50, 51) : intervalle angulaire

– Une liste des arcs à dessiner

Les polylignes sont « coupées » au niveau des sommets reliant deux segments (i.e. qui ne sont pas des extrémités) pour les décomposer en lignes.

Initialisation de la carte

On déduit des informations de construction, retournées par le pré-traitement, les dimensions de la carte en calculant la différence entre les coordonnées du coin haut droit et celles du coin bas gauche. Ces dimensions qu'on calcule ainsi sont alors exprimées dans l'unité définie au *HEADER* du fichier *DXF* original. Il faut alors les convertir en mètres. Ce qui est effectué connaissant l'unité qui fait également partie des informations de construction.
Finalement, une carte est initialisée avec une grille d'occupation de largeur et de hauteur calculées par la division des dimensions en mètres par la résolution (i.e. la taille d'une cellule). Toutes les cellules de cette grille sont libres à ce niveau et il est temps de commencer à dessiner les obstacles, ou encore à déterminer quelles sont les cellules occupées.

Construction par élément

La dernière étape consiste à dessiner tous les objets collectés dans des listes sur la carte. On dessine ces objets un par un en utilisant la méthode suivante pour chaque objet :

– D'abord, on localise l'objet dans la carte. Cette localisation passe par le calcul de la différence entre les coordonnées des points principaux de l'élément graphique et celles du coin bas gauche, et ensuite, par la conversion des positions en mètre puis en nombre de cellules.
– Ensuite, on définit la zone locale de construction.
– Puis, on rectifie les valeurs des cellules occupées pour dessiner l'objet.

Construction de lignes : La zone de construction d'une ligne est définie à partir de ses extrémités dont les coordonnées sont converties en nombre de cellules comme le montre la Figure 4.4.
Pour vérifier si une cellule de la zone de traitement est sur la ligne, donc occupée, on utilise la propriété de *Cauchy-Shwarz* qui stipule que pour tout vecteurs x et y du plan :

$$|\langle x|y\rangle| \leq \|x\|\|y\|$$

où $\langle .|.\rangle$ désigne le produit scalaire et $\|.\|$ la norme euclidienne.
L'égalité est vérifiée uniquement dans le cas de colinéarité des deux vecteurs.
Ainsi, si on pose une cellule $C(i,j)$ [3] de la zone de dessin d'une ligne, elle est considérée occupée si elle vérifie :

$$\|\overrightarrow{D_{pxl}C}\|\|\overrightarrow{D_{pxl}F_{pxl}}\| - |\langle\overrightarrow{D_{pxl}C}|\overrightarrow{D_{pxl}F_{pxl}}\rangle| < \epsilon_l$$

où ϵ_l est une constante positive choisie empiriquement.

Construction des cercles : Dans le cas d'un cercle, la zone de traitement est une fenêtre carrée définie à partir du centre du cercle O_{pxl} et de son rayon (converti en nombre de cellules) R_{pxl} à la manière décrite par la Figure 4.5.
Dans ce cas, pour vérifier si une cellule $C(i,j)$ est sur le cercle, il suffit de vérifier que :

$$R^2 - \epsilon_c < OC^2 < R^2 + \epsilon_c$$

où ϵ_c est une constante positive choisie empiriquement et où OC^2 est définie par :

$$OC^2 = [(i - O^i_{pxl}).resolution]^2 + [(j - O^j_{pxl}).resolution]^2 \tag{4.1}$$

avec $O_{pxl}(O^i_{pxl}, O^j_{pxl})$.

Construction des arcs : Soit $O_{pxl}(O^i_{pxl}, O^j_{pxl})$ le centre de l'arc exprimé en nombre de cellules, $\phi_1 < \phi_2$ les angles qui le délimitent et R son rayon.
La zone de dessin de l'arc dans la grille de cellules est la même que celle définie en supposant que nous avons un cercle.

3. on assimile une cellule à un point de coordonnées le numéro de colonne et le numéro de ligne dans la grille.

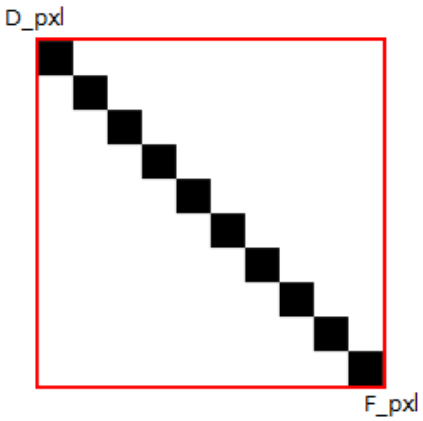

Figure 4.4 Zone de dessin d'une ligne

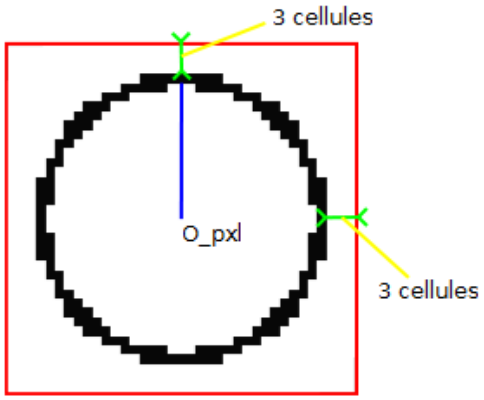

Figure 4.5 Zone de dessin d'un cercle

Une cellule $C(i, j)$ de cette zone est sur l'arc si elle vérifie :

$$R^2 - \epsilon_a < OC^2 < R^2 + \epsilon_a$$
$$\phi_1 \leq atan2(j.resolution, i.resolution) \leq \phi_2$$

où ϵ_a est une constante positive choisie empiriquement et où OC^2 est définie par (4.1).

4.3 Programme et Application

4.3.1 Éléments importants de l'implémentation

Le programme tel qu'il a été conçu (voir 4.2.2) comporte deux grandes parties qui présentent des particularités touchant directement à l'implémentation.
D'une part, « l'échange de fichier » entre les deux parties décrit par la structure se fait pratiquement par échange du chemin d'accès du fichier. Les informations de construction peuvent également être échangées par la même voie.
D'autre part, le composant de pré-traitement est clairement purement indépendant des outils du système d'exploitation robotique utilisé pour la programmation du système de navigation du fauteuil. Il ne dépend que d'outils généraux et classiques de lecture et d'écriture de fichier.

Ainsi, ces deux points nous permettent de conclure que nous ne sommes pas obligés d'implémenter les deux modules en un même environnement. Ils offrent même la possibilité d'avoir un premier composant portable et programmé sous un autre langage que le deuxième qui, lui par contre, doit forcément se présenter sous la forme d'un nœud [4] de *ROS* puisqu'il

4. un nœud est un module communiquant avec d'autres dans le réseau du système en *ROS*. Consulter O'Kane (2013) pour plus d'informations.

publie la carte en grille d'occupation construite dans le réseau du FRMI.

Nous avons alors profité de cette possibilité pour optimiser la vitesse de la phase de pré-traitement et pour alléger sa programmation. Elle est du coup implémentée sous *Matlab*. On assure par ce choix sa portabilité, puisque nous n'aurons besoin que des scripts *Matlab* pour la lancer, et sa flexibilité, vu la facilité de correction directe sur le code offerte par *Matlab*.

4.3.2 Exemples de cartes d'étages construites

On se propose de tester la technique de génération automatique de cartes à partir de plans architecturaux présentée dans ce chapitre.
On effectue alors la génération des cartes de 3 étages du pavillon Lassonde de l'École Polytechnique : Étages 1, 3 et 5.

Les plans de ces étages fournis par l'administration de l'École Polytechnique sont initialement sous le format DWG. A l'aide du logiciel de CAO *AutoCAD*, on a pu convertir sans difficultés les 3 plans d'étage au format *DXF*. On est ainsi prêt sans avoir à faire une autre manipulation à générer les cartes complètes nécessaires à la navigation du FRMI dans trois étages ! Sans même avoir à se rendre sur les lieux de l'éventuelle navigation prévue !

Dans ce qui suit, nous allons présenter les résultats de tests pour chaque étage un par un. Seulement les résultats relatifs à l'étage 5, qui est le premier présenté, montreront les outputs et la manipulation avec le maximum de détails. Pour les autres étages, nous nous limitons du coup à la présentation du plan original et de la carte construite automatiquement dans chaque cas.

Génération de la carte de l'étage 5

Le plan original du cinquième étage du pavillon *Lassonde* de Polytechnique est montré à la figure 4.6. On voit clairement un énorme tas de détails supplémentaires ne faisant pas partie des obstacles que nous cherchons à séparer du reste des objets : on peut voir des écritures, des axes en traits rouges fins, deux échelles et un titre en jaune, des numéros, etc.

La première phase de pré-traitement nous permet d'avoir la liste complète des calques du fichier *DXF* du plan. Un examen rapide de cette liste nous permet immédiatement d'avoir une idée de la liste de calques à choisir parmi tous. La figure 4.7 présente ainsi un exemple de compilation de calques choisis.
En effet, on garde les noms de calques de terminaisons du genre : **MUR**, **FEN** (pour fenêtre), **ESC** (pour escaliers), **COL** (pour colonne), ... On garde également les noms contenant l'expression **BET** (pour béton). Et bien évidemment , on veille en contre partie à éliminer les noms de terminaisons du genre **AXE** ...

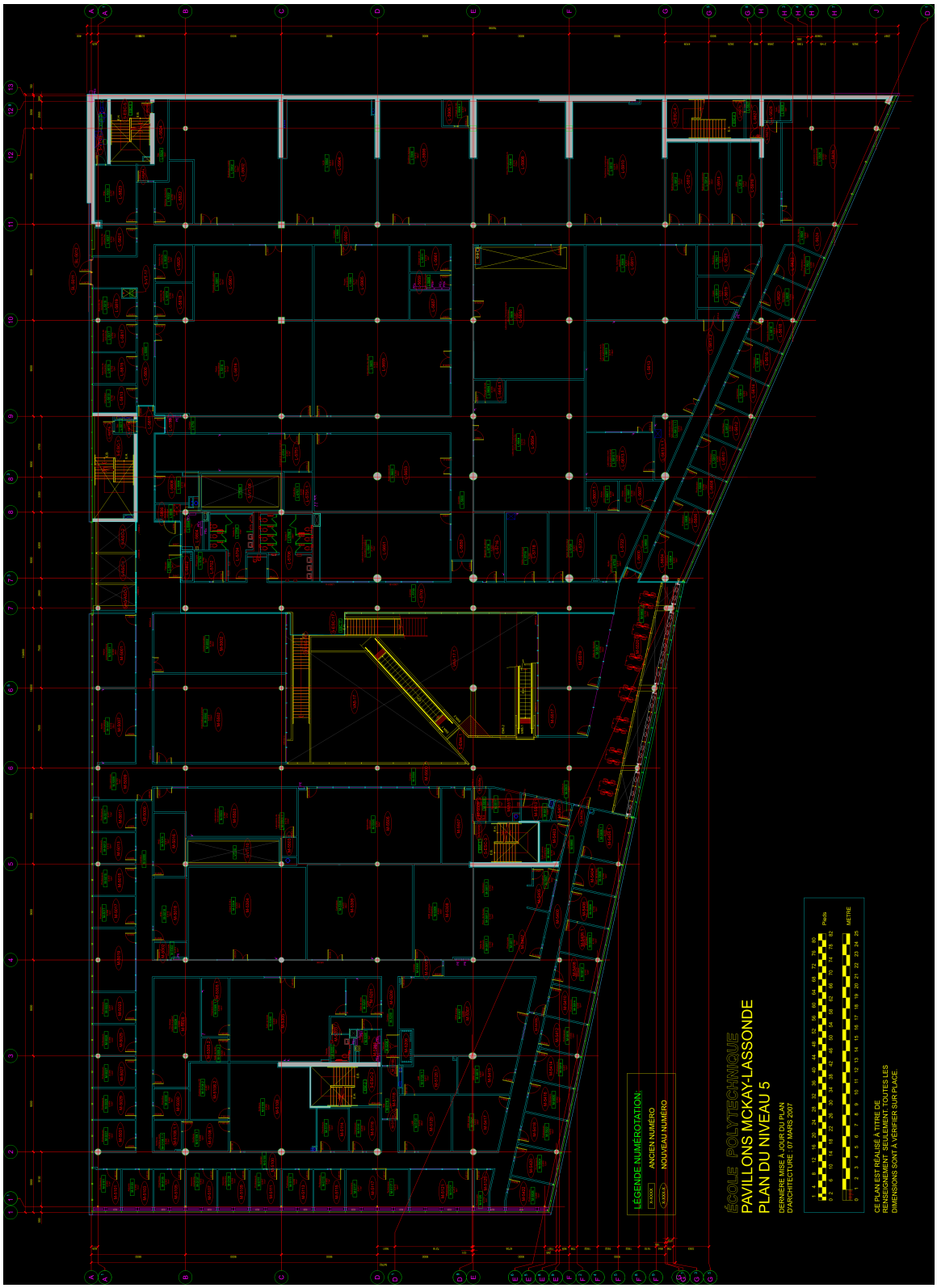

Figure 4.6 Visualisation du plan d'architecture original du cinquème étage du pavillon Lassonde de Polytechnique

Notons que ce choix est totalement intuitif et ne faisant appel à aucune règle précise. Surtout si nous savons que les noms de calques sont le fruit du choix du dessinateur du plan original et peuvent respecter des conventions (ou pas) différentes d'un plan à un autre. Mais, du moment que l'utilisateur de notre programme a la possibilité de rectifier ce choix autant de fois qu'il ne le veut, une première compilation de calques nettement sous-optimale n'est donc pas grave.

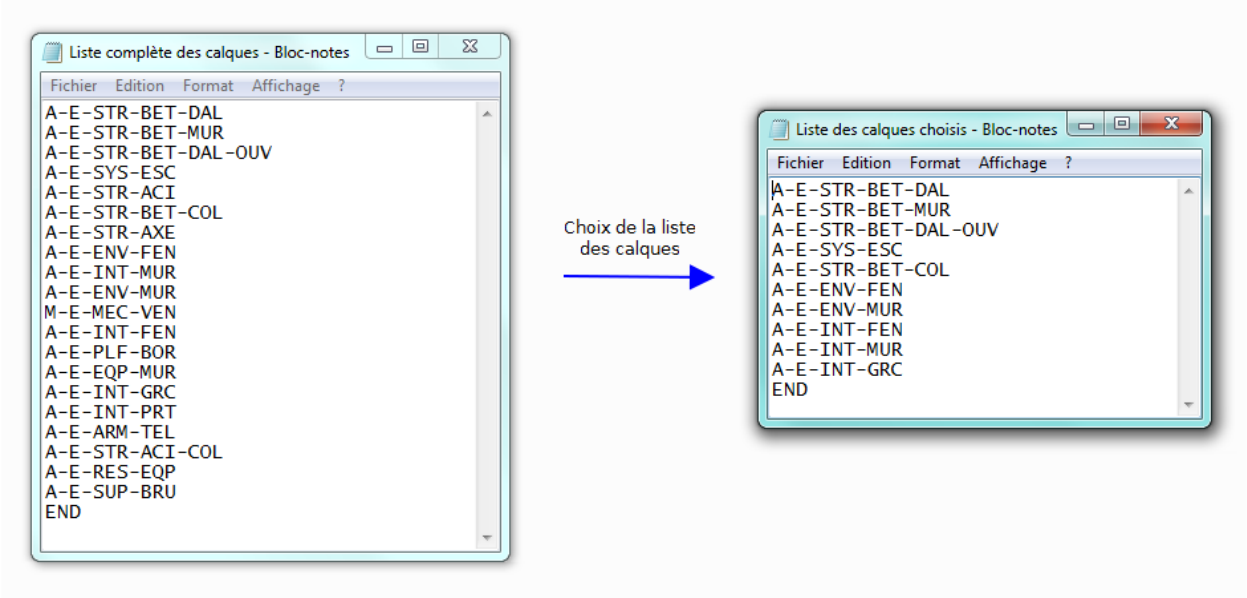

Figure 4.7 Liste complète des calques du plan de l'étage 5 et liste des calques choisis

La visualisation du plan simplifié résultant de la liste des calques que nous choisissons (Figure 4.8) confirme la logique de notre choix. Il est possible à ce niveau d'améliorer encore la liste des calques, comme expliqué plus haut, pour essayer d'assurer plus d'optimalité. Il s'agit d'un compromis à faire cherchant à avoir le meilleur plan simplifié. Donc, celui qui renferme le minimum d'objets « virtuels » et le maximum d'obstacles utiles pour la navigation.
Précisons finalement que le plan simplifié construit utilise un seul calque et un seul type de trait (standard) et élimine du coup toute option et toute couleur du plan original.

Enfin, comme nous sommes satisfaits de notre plan simplifié, on lance sa construction automatique pour aboutir à la publication de la grille d'occupation correspondante dans le réseau du système du FRMI. Le résultat obtenu est donné par la figure 4.9.

En quelques minutes, on a pu construire une carte presque complète d'un étage entier. Cette même carte aurait nécessité des heures de navigation (et donc des heures de consommation de la charge, et donc des heures d'usage de batterie ! Et des heures de perdues de la durée de vie des capteurs et des joints du FRMI !) si nous avions pris la décision d'aller explorer

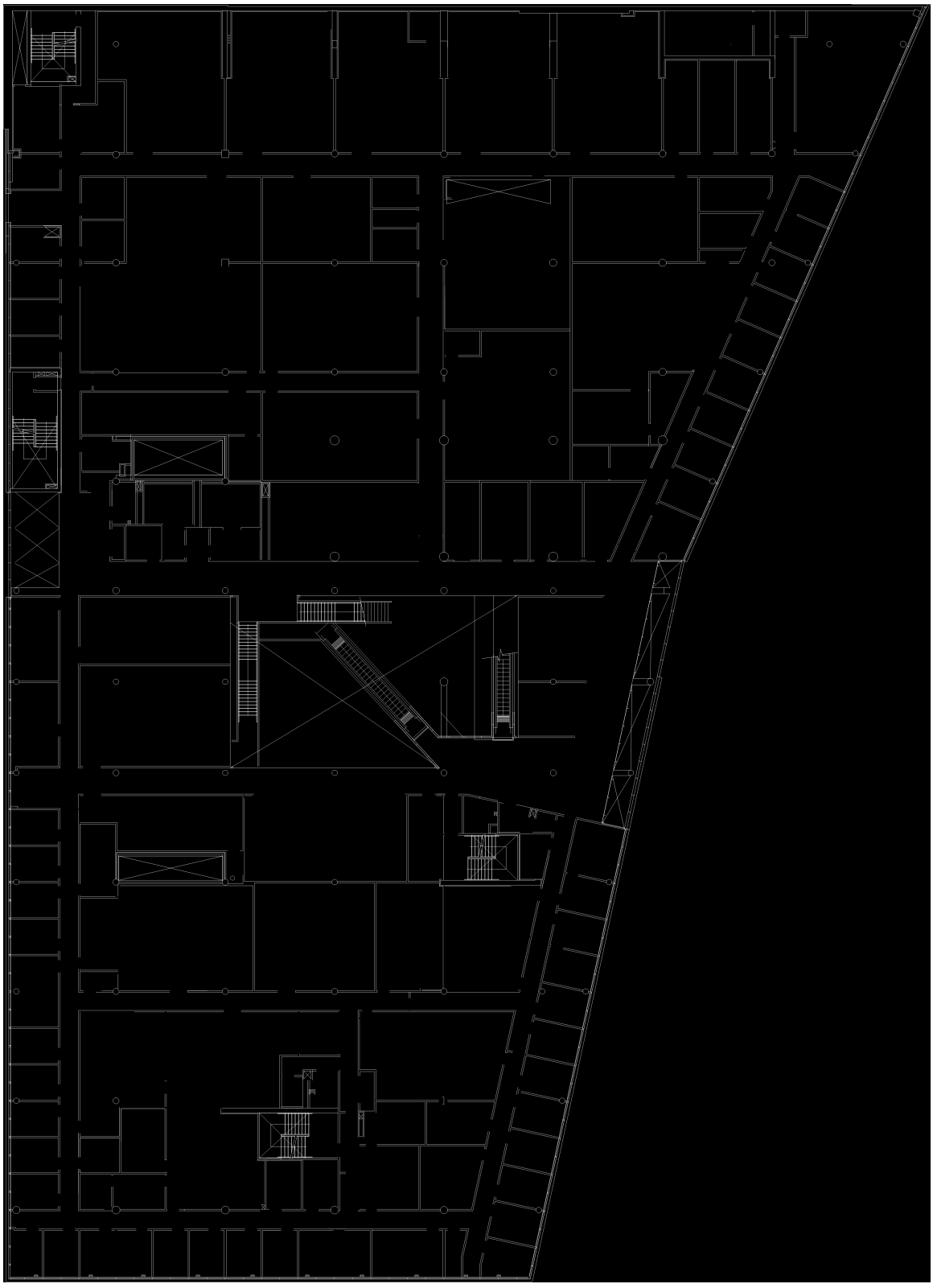

Figure 4.8 Plan simplifié de l'étage 5

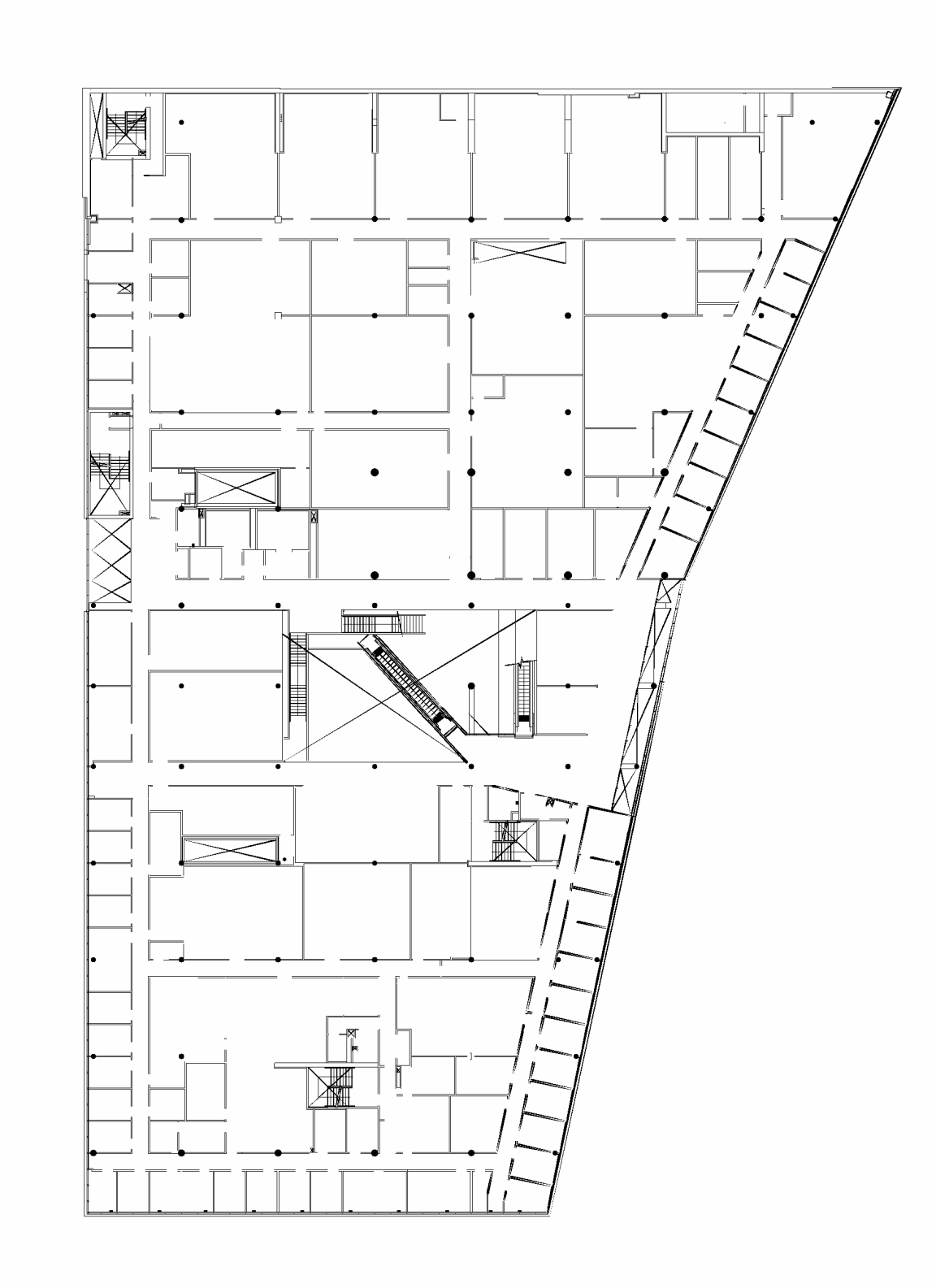

Figure 4.9 Carte de navigation de l'étage 5 générée par notre programme avec une résolution de 0.05m

tous les recoins de l'étage, avec un module de cartographie par SLAM (tel que le GMapping) lancé sans interruption et les télémètres laser en marche, pour la construire. Par exemple, la construction de la carte de la figure 3.16 a pris environ 45 minutes pour ne représenter que moins de 20% de la surface à explorer. Tandis que notre module de génération automatique n'a pris que 5 minutes environ (La phase de pré-traitement varie car elle nécessite l'intervention de l'utilisateur du programme) pour donner la carte 4.9
Et même en effectuant cette lourde expérience de cartographie par SLAM, l'obtention d'une carte aussi consistante et précise est loin d'être à la portée du FRMI (ou de n'importe quel robot doté d'un module de cartographie par SLAM) vu que des difficultés, tel que la fermeture d'une boucle en reconnaissant un endroit par lequel le robot est déjà passé, constituent encore un grand défi à la cartographie robotique par SLAM. Rappelons en guise d'illustration la carte de la figure 2.6 construite à l'aide du *GMapping*.

Génération des cartes des premier et troisième étages

Nous fournissons les plans originaux accompagnés des résultats de construction automatique des cartes de deux autres étages dans les figures 4.10, 4.11, 4.12 et 4.13. Ces deux étages sont de structure (du point de vue répartition des obstacles et des espaces libres) différente par rapport à l'étage 5. En particulier, le premier étage montre des couloirs plus large formant un cycle qui entoure de très larges pièces (amphis) en plus d'un secteur isolé (section accessible seulement pour le personnel de Polytechnique).
Les cartes obtenues sont assez complètes et montrent les obstacles essentiels ainsi que les espaces navigables. On remarque cependant que certains murs manquent par endroit. Ceci est dû à la mauvaise répartition des éléments graphiques sur les calques du plan original ce qui nous oblige à sacrifier certains obstacles pour ne pas en créer beaucoup d'autres supplémentaires. D'ailleurs, même la carte de l'étage 5 à la figure 4.9 a par endroit des obstacles manquants.
Ces défauts ne devraient généralement pas altérer la navigation du FRMI. En effet, le fauteuil se localise par rapport à la carte en utilisant un module basé sur une estimation probabiliste de la position d'un ensemble de particules (Ce module est exposé avec plus de détails à la partie 4.4 suivante). Ce recours à une méthode d'estimation probabiliste est tout a fait normal parce que même si nous parvenons à construire une grille d'occupation parfaite où figurent tous les obstacles « statiques », il y aura toujours des poubelles, des tables, des chariots, des passants et toute sorte d'obstacles « dynamiques ». Du coup, la position la plus probable calculée par comparaison des observations des télémètres laser aux cellules occupées constitue le moyen du FRMI pour estimer sa position avec tout ce « bruit » inévitable.
La partie 4.4 montre des exemples de tests qui illustrent l'efficacité de la localisation dans la

carte du cinquième étage.
Elle montre également des résultats d'utilisation de notre module d'enrichissement du chapitre 3 pour compléter cette carte, ce qui représente une solution efficace aux défauts évoqués ici.

4.4 Utilisation de la carte construite en navigation

Pour conclure, nous nous proposons de montrer des exemples d'utilisation d'une carte construite automatiquement dans le but de prouver l'utilité de la technique présentée dans ce chapitre à la navigation globale.
Ainsi, nous présentons d'abord l'efficacité de la localisation du FRMI dans la grille d'occupation de l'étage 5 pour mettre le point sur la précision des dimensions de celle-ci.
Ensuite, nous allons présenter des exemples de détection d'obstacles et de navigation point à point réussis.
Enfin, nous allons pousser l'exploitation de notre carte construite à un niveau supérieur en testant notre technique d'enrichissement de carte exposée au chapitre 3 précédent pour compléter correctement et avec précision certains obstacles non représentés sur la carte générée à partir du plan d'architecture.

4.4.1 Navigation manuelle et auto-localisation

Pour profiter d'une carte de l'environnement, il faut que le FRMI soit capable de « naviguer dans cette carte ». « Naviguer dans la carte » signifie que l'utilisateur du fauteuil peut se déplacer dans le milieu réel cartographié tout en garantissant que le FRMI se localise, en parallèle, par rapport à la carte en mémoire avec précision.
Pour se localiser, un robot peut utiliser son odométrie. Cependant, l'odométrie n'est pas suffisante pour garder une estimation exacte de la position par rapport à la carte car les erreurs (dues essentiellement aux glissements des roues) sont cumulatives. Ces erreurs sont d'autant plus flagrantes en rotation. D'où la nécessité d'un module de localisation qui utilise les observations du FRMI retournés par ses télémètres laser. Ces derniers retournent des informations sur les obstacles qui entourent le fauteuil qui sont alors comparées aux informations de la carte et c'est là que notre carte construite prend toute son utilité pour la navigation globale.

Ici, notre objectif n'est pas de prouver l'efficacité de la localisation ni de l'étudier mais de mettre le point sur l'importance et l'utilisation de la carte construite automatiquement sur deux niveaux :

- la precision de sa construction et l'exactitude de ses dimensions
- son intégration dans le système et son utilisation pour l'autolocalisation du FRMI

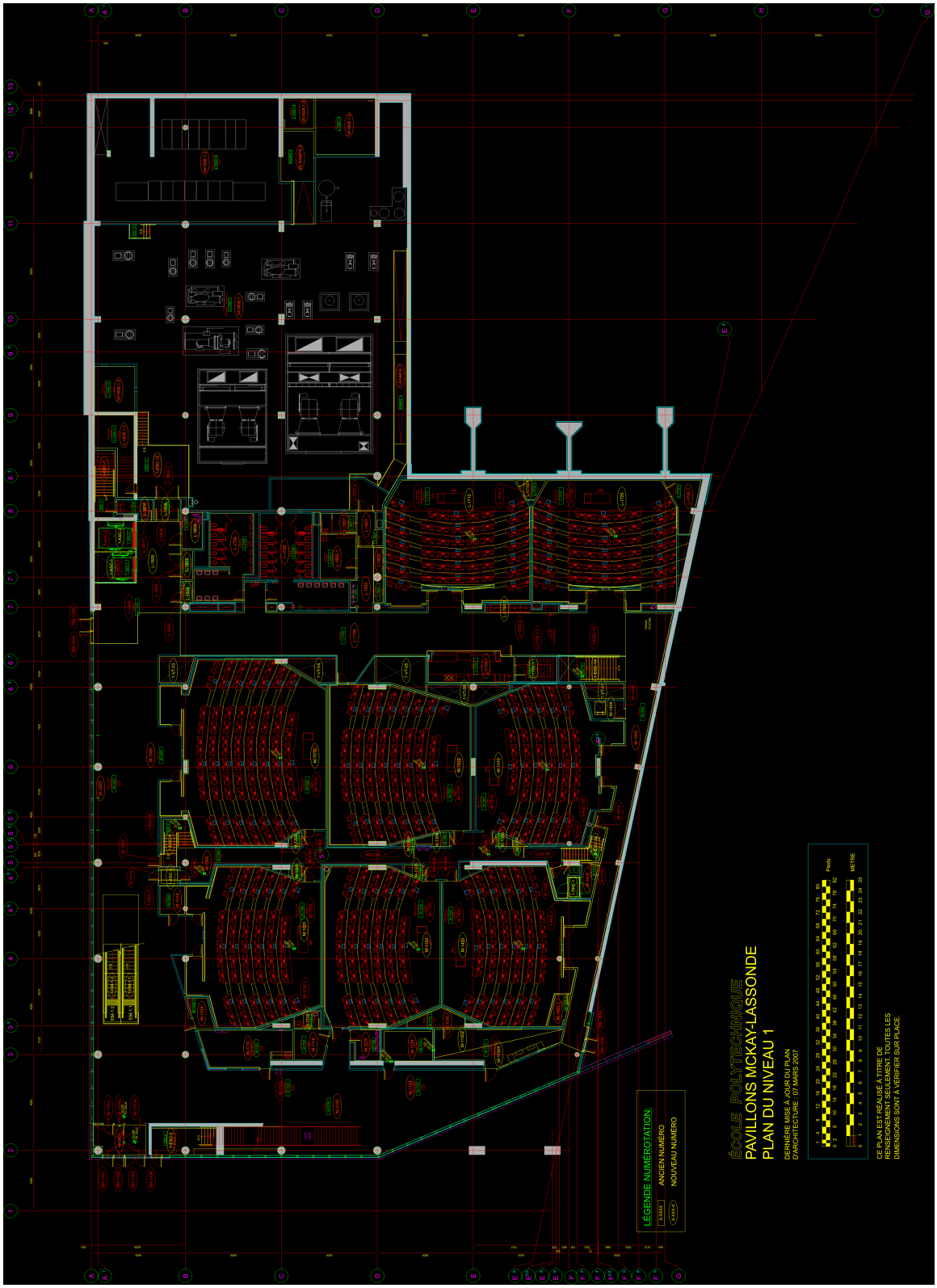

Figure 4.10 Visualisation du plan d'architecture original du premier étage du pavillon Lassonde de Polytechnique

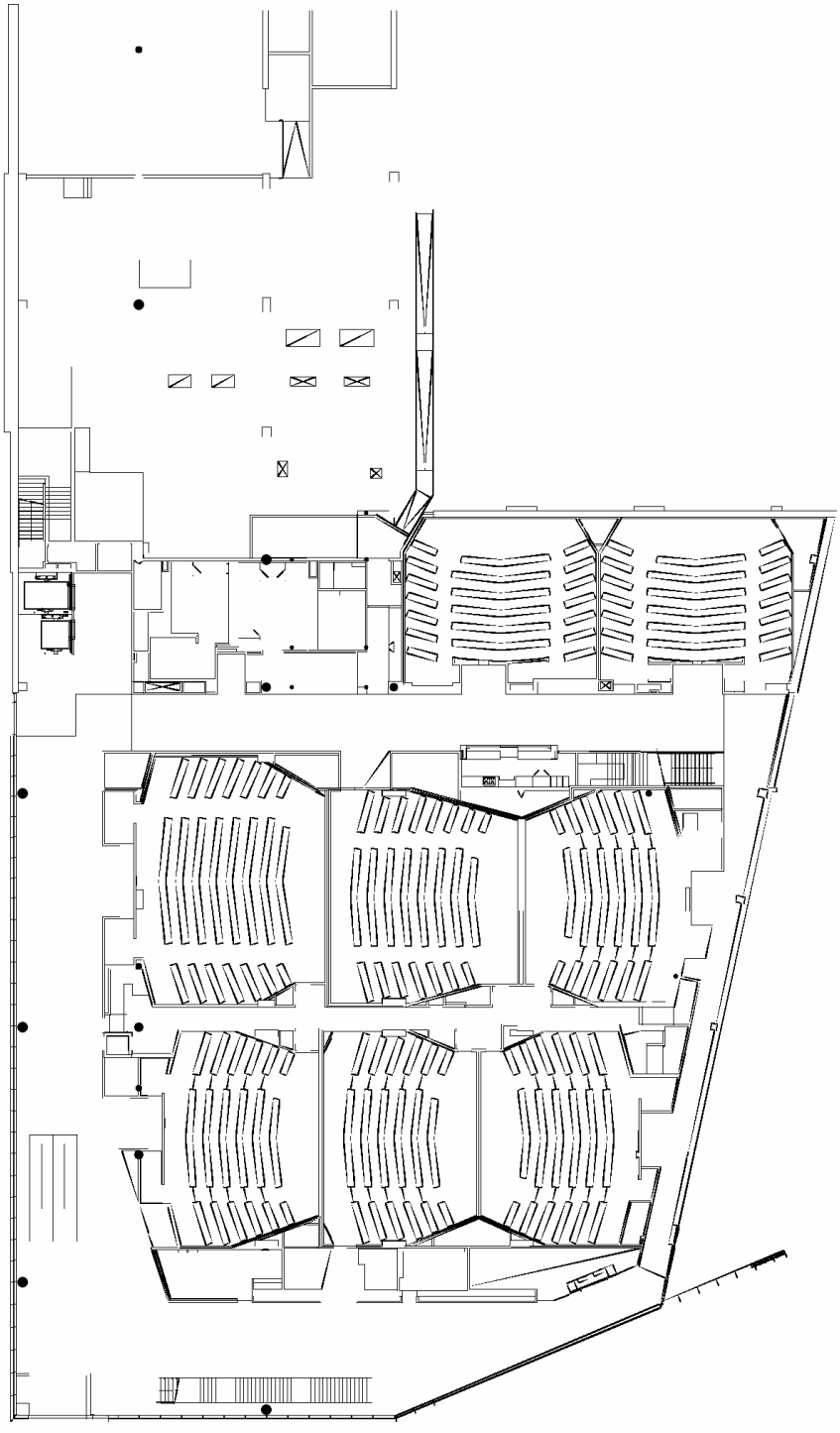

Figure 4.11 Carte de navigation du premier étage générée par notre programme avec une résolution de 0.05m

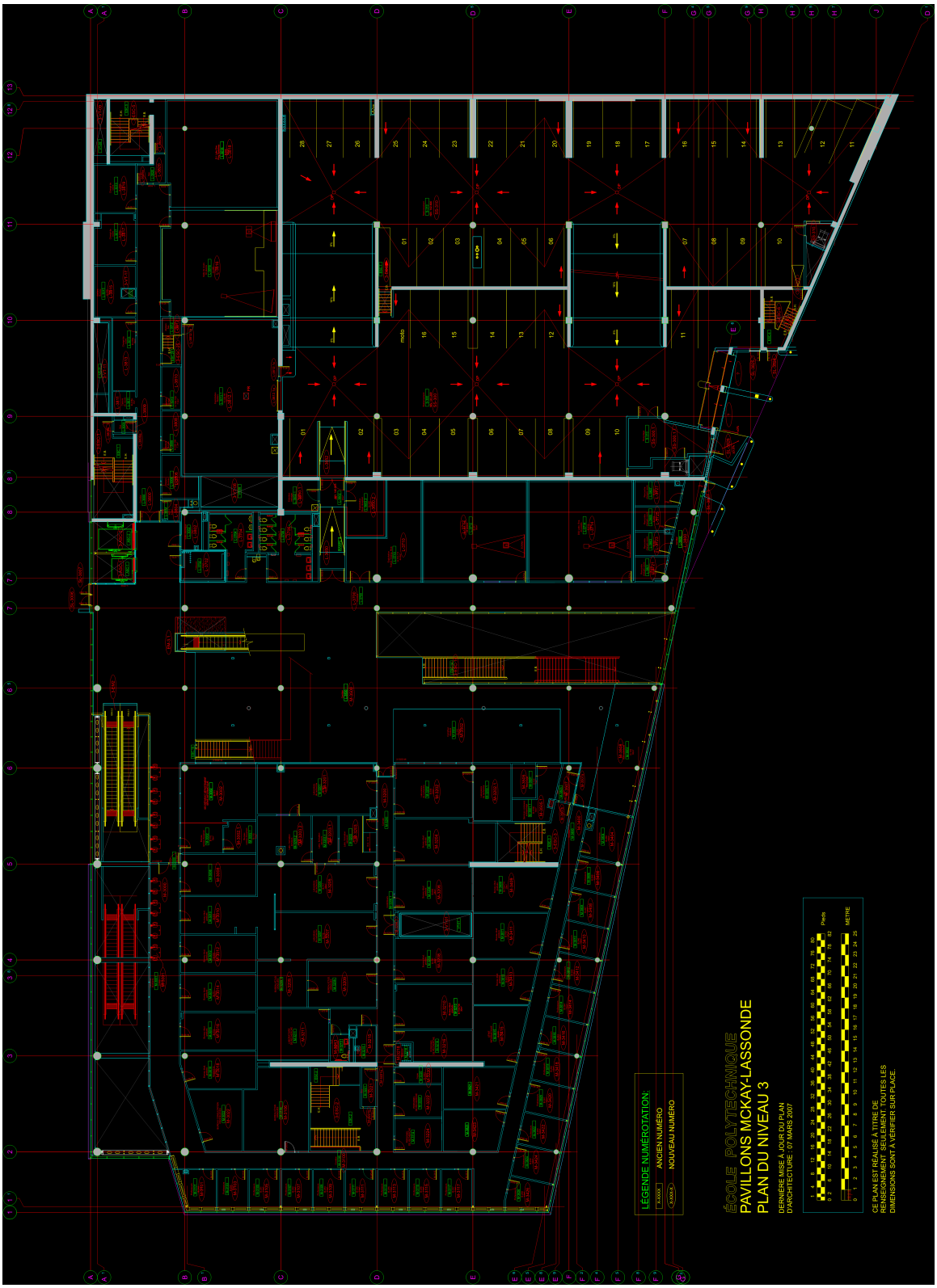

Figure 4.12 Visualisation du plan d'architecture original du troixième étage du pavillon Lassonde de Polytechnique

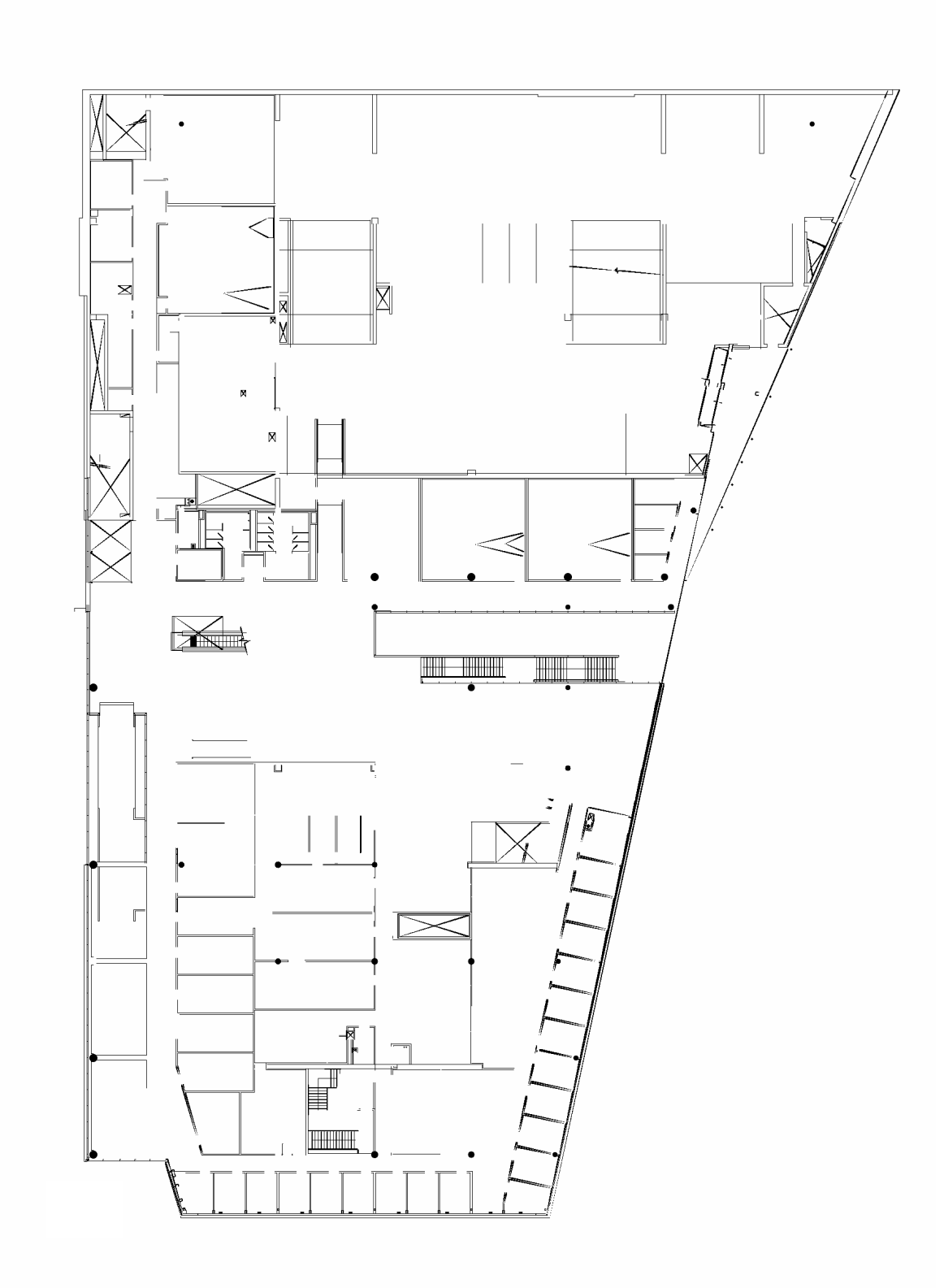

Figure 4.13 Carte de navigation du troixème étage générée par notre programme avec une résolution de 0.05m

Pour cela, nous exposons ci-après les captures de deux expériences de navigation manuelle dans la carte construite du cinquième étage.
La figure 4.14 montre les emplacements des captures dans la carte globale.
Dans les deux expériences, nous effectuons une simple navigation manuelle avec une auto-localisation active.

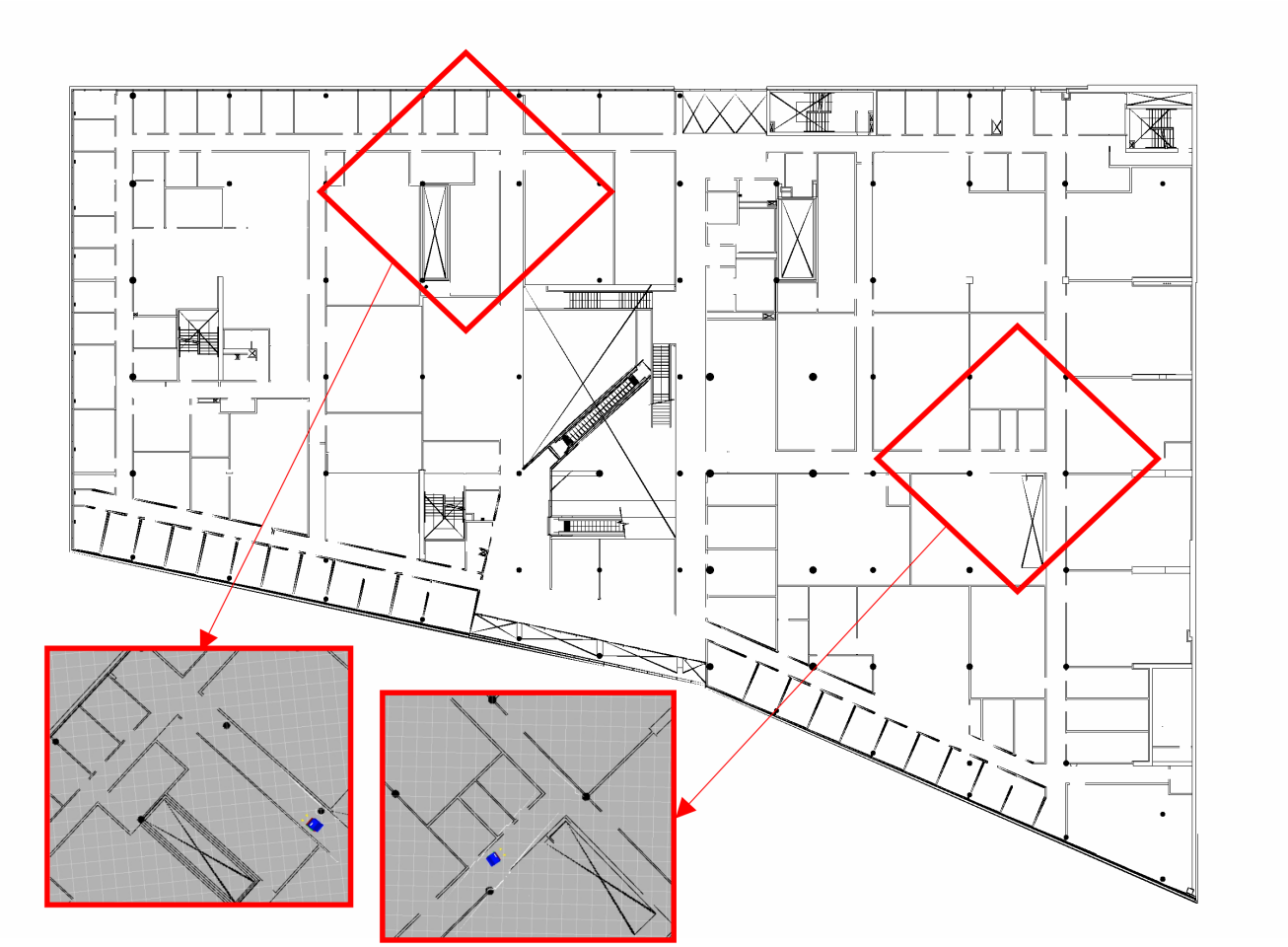

Figure 4.14 Secteurs de test de la navigation manuelle dans la carte générée du cinquième étage

La première expérience (voir Figure 4.15) sert essentiellement à montrer l'exactitude des dimensions de la carte construite. On bouge alors le FRMI dans un couloir étroit vers un cul-de-sac, tout en évitant une colonne sur le chemin. Les résultats montrent sur toutes les captures que les détections laser (traces blanches) sont sensiblement compatibles avec les obstacles (en noir) de la carte. Vers le cul-de-sac, on remarque la présence d'obstacles non représentées dans la carte placées contre le mur (des armoires et casiers sur les côtés à la

capture E). Mais, la ligne blanche marquant la détection du fond du cul-de-sac confirme la précision de construction. Les erreurs translationnelle et angulaire maximales mesurées sont respectivement égales à $\underline{2cm}$ et $\underline{0.09rad}$.

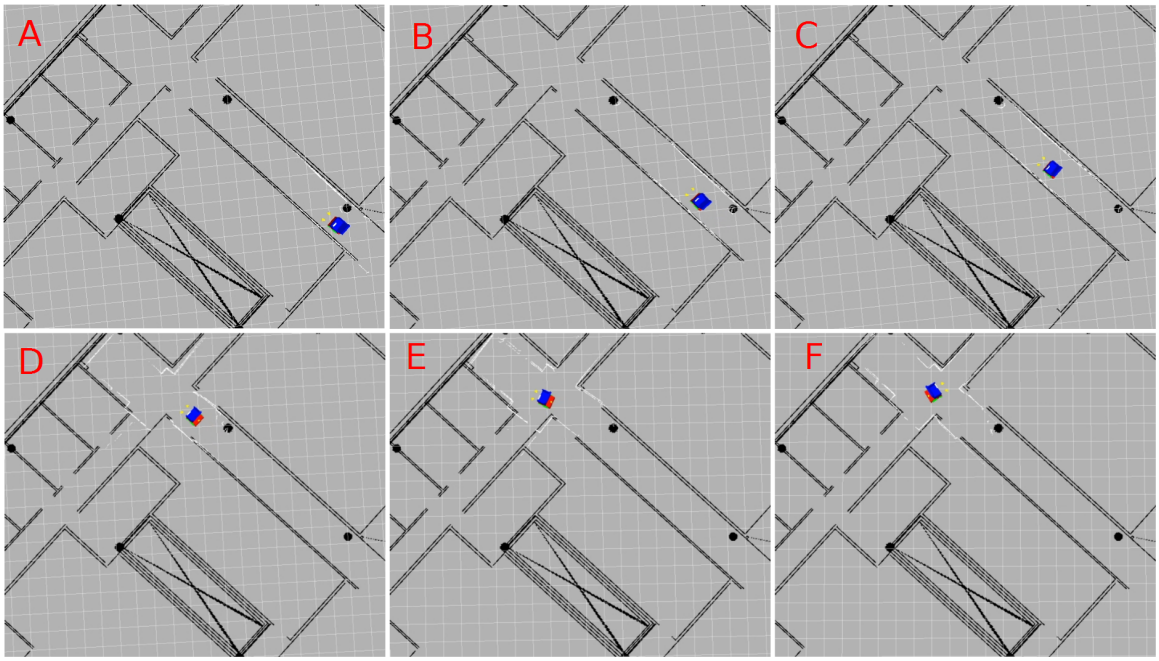

Figure 4.15 Navigation manuelle avec auto-localisation dans une carte générée automatiquement : position initiale sans décalage

En deuxième expérience, on teste la correction de la localisation. On démarre alors l'expérience avec un décalage angulaire (voir capture A de la figure 4.16). A cause du décalage, on voit clairement que les détections des laser ratent complètement la colonne vers le fond du couloir de départ sur la carte. Au fil de la navigation, l'erreur est corrigée progressivement (entre C et E). Par contre, la rotation au croisement pour prendre le couloir perpendiculaire vers la droite crée un autre léger décalage (dans l'autre sens cette fois-ci) visible à la capture F qui est à son tour rapidement corrigé.

Le module que nous utilisons pour l'auto-localisation est le module AMCL [5] de ROS. L'AMCL applique l'algorithme probabiliste décrit par Fox (2001) qui est basé sur un filtre particulaire adaptif i.e. de taille d'échantillon intelligemment variable selon la dispersion des particules. En effet, en cas de grande dispersion (la localisation est difficile) due à une répartition des particules sur une plus grande surface de la carte et, donc, traduisant une faible densité de particules, la taille de l'échantillon est augmentée afin d'augmenter la densité (pour faciliter l'estimation de la position) et vice versa. Bien évidemment, dans une grande carte

5. Adaptive Monte Carlo Localization

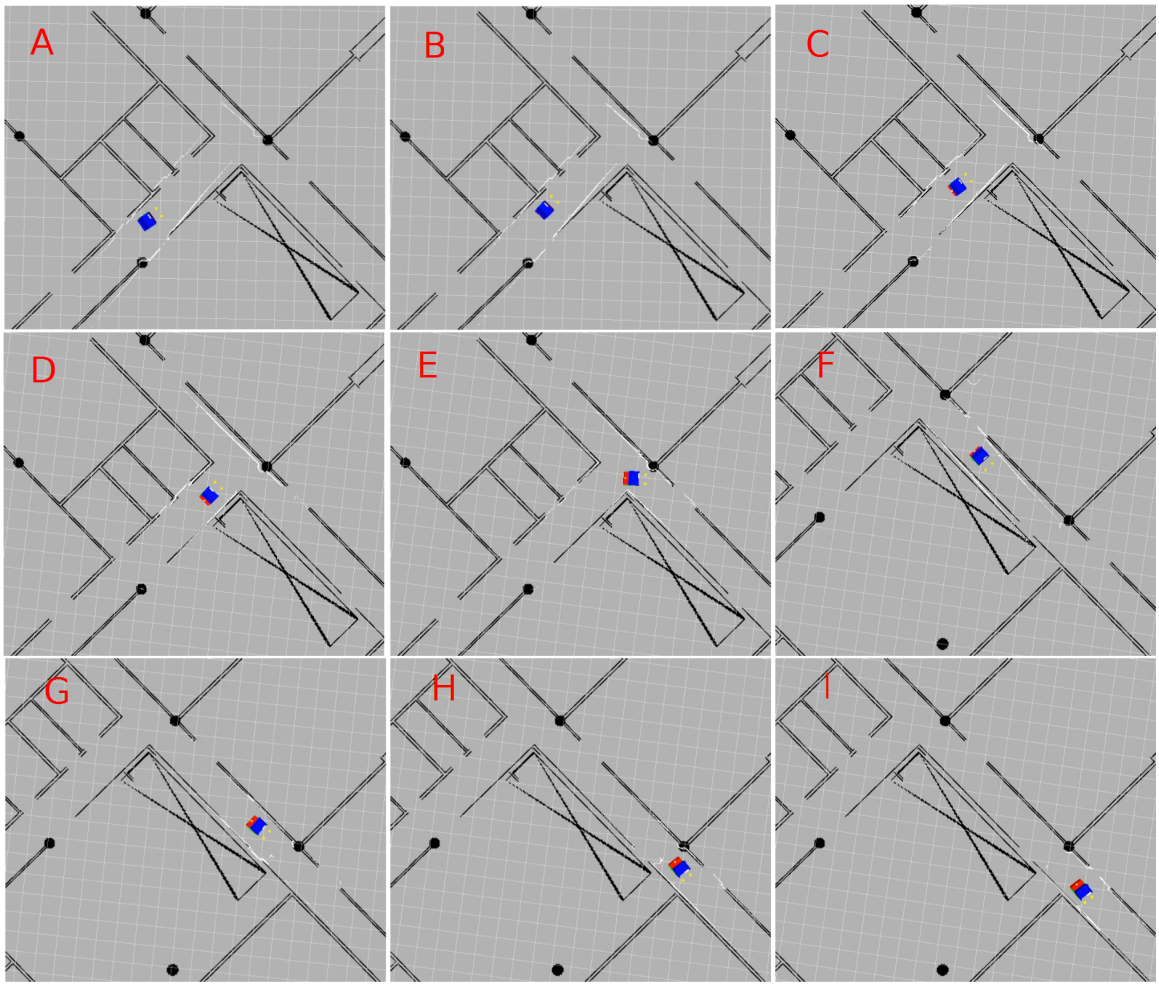

Figure 4.16 Navigation manuelle avec auto-localisation dans une carte générée automatiquement : position initiale avec décalage

avec un niveau de symétrie élevé, la localisation est impossible sans choix d'une pose initiale de position et d'orientation suffisamment proches de la pose exacte. L'exemple classique d'un long couloir symétrique que l'on parcourt dans un sens ou dans l'autre donne un cas où la localisation ne peut pas différencier les deux sens.

4.4.2 Navigation point à point dans la carte

Si on a un robot qui se localise correctement dans notre carte construite automatiquement à partir d'un plan CAO, la prochaine étape serait de vouloir passer d'une navigation manuelle à une navigation automatique.
Nous cherchons alors maintenant à tester la navigation point à point dans notre carte du cinquième étage. Effectuer cette tâche correctement passe, en plus de la localisation, par la détection des obstacles (y compris ceux qui ne sont pas dessinés sur la carte construite) pour planifier un chemin correct.

Nous effectuons donc les deux expériences schématisées par la figure 4.17 où l'on donne les captures des poses initiales du FRMI dans la carte ainsi que les chemins suivis globalement par le FRMI tracés en vert.
Pour chaque expérience, l'auto-localisation par AMCL est en marche et on utilise le module *move_base* de ROS. Ce module crée (en faisant appel à plusieurs nœuds liés [6]) à partir de notre carte « statique » construite, une carte dynamique montrant les obstacles à la portée des télémètres laser de la chaise. Ces obstacles sont enflés (voir tâches rouges sur les captures) sur cette carte qui est mise à jour régulièrement dans le but d'assurer une distance de sécurité entre le FRMI et les obstacles qui peuvent représenter un risque de collision.
Le module reçoit nos poses finales choisies dans la carte, et donc exprimés dans le repère *Monde*. Il commence ainsi la planification de chemin et suit le trajet planifié tout en évitant les obstacles locaux. En cas de besoin, le robot rectifie son chemin global pour contourner un obstacle non détecté au départ.

Les captures des deux expériences réalisées dans ce cadre sont données à la figure 4.18. Ces deux exemples illustrent le succès de l'intégration de la carte résultante de la technique présentée dans ce chapitre dans le système de navigation automatique sur plusieurs niveaux :

- La carte sert de support à la création de la grille d'occupation mise à jour dynamiquement pour montrer les obstacles enflés qui entourent le FRMI.
- La pose où nous voulons que le FRMI se rende est choisie sur la carte, la planification de chemin qui en résulte est correcte.
- La localisation dans la carte fournit le feed-back nécessaire pour générer la commande permettant de garder le FRMI sur son chemin.

6. voir la documentation de *move_base* sur le wiki de ROS

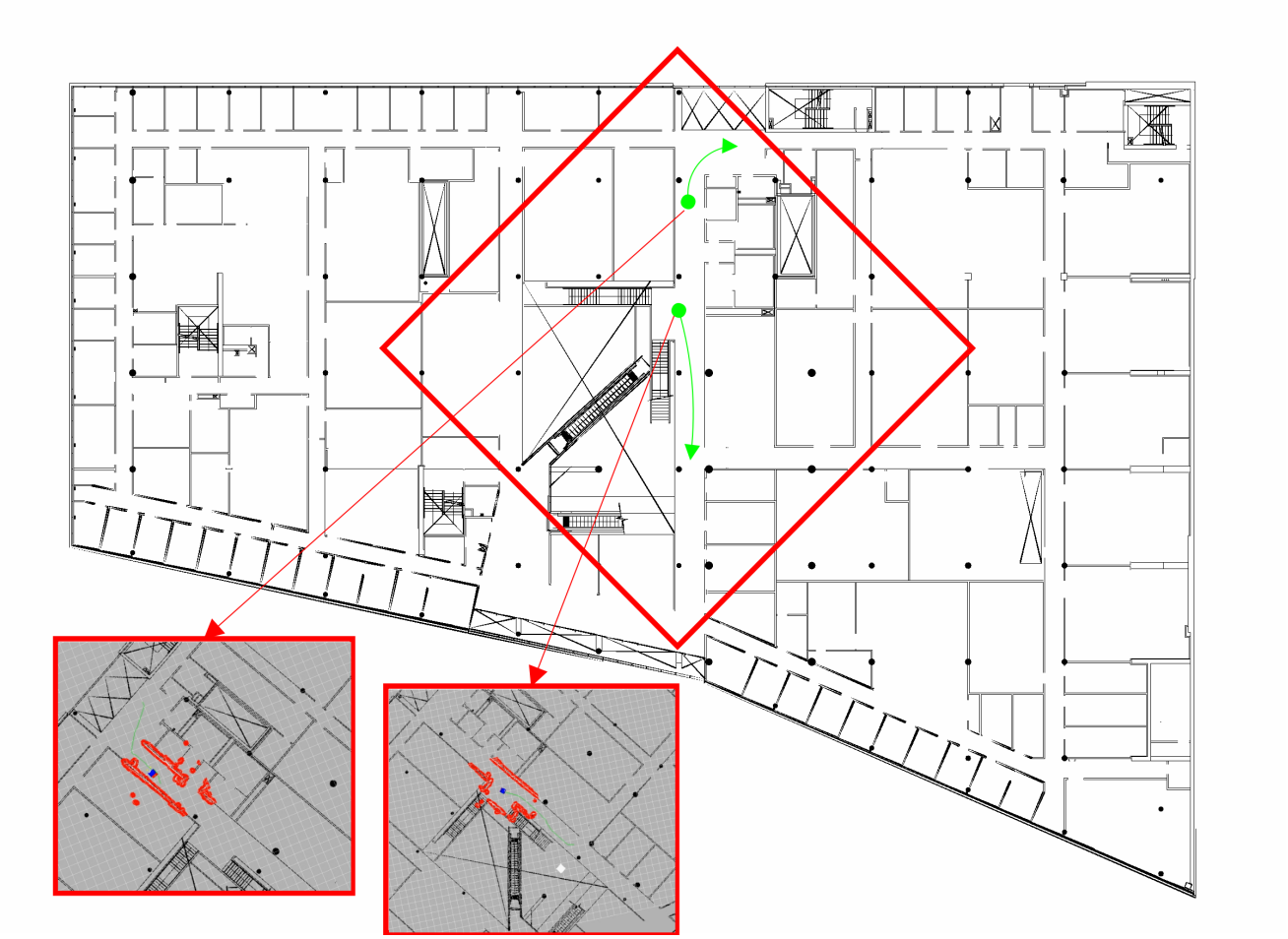

Figure 4.17 Schéma des tests de la navigation point à point dans la carte construite du cinquième étage du pavillon Lassonde

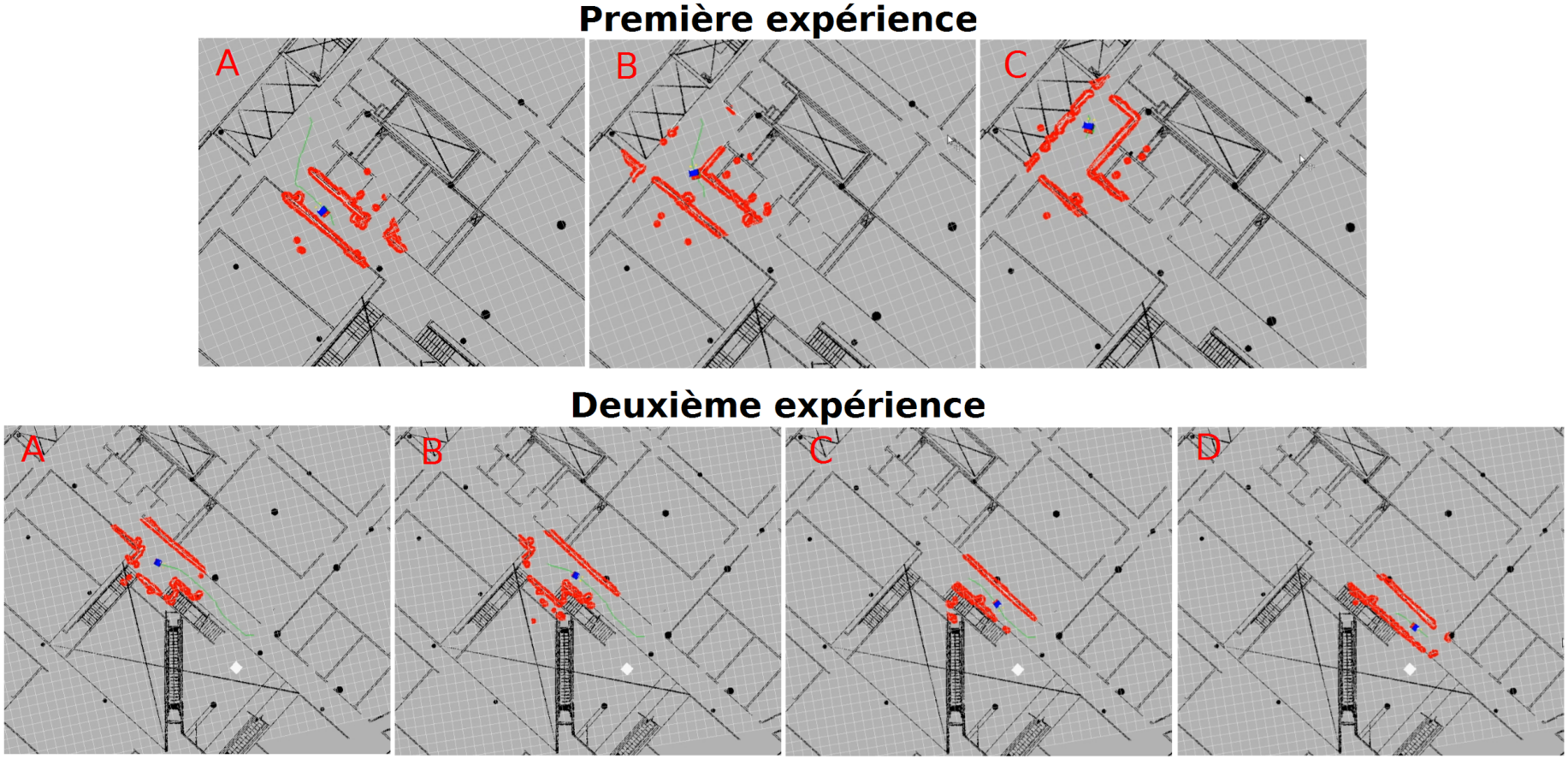

Figure 4.18 Captures des tests de la navigation point à point dans la carte construite du cinquième étage du pavillon Lassonde

4.4.3 Enrichissement intelligent par les détections des télémètres ultrason

Nous avons mentionné en 4.3.2 que la carte construite automatiquement par notre programme peut ne pas contenir tous les obstacles statiques. Tel est le cas de la carte du cinquième étage que nous avons générée. Nous montrons deux exemples de ces obstacles manquants sur la figure 4.19.

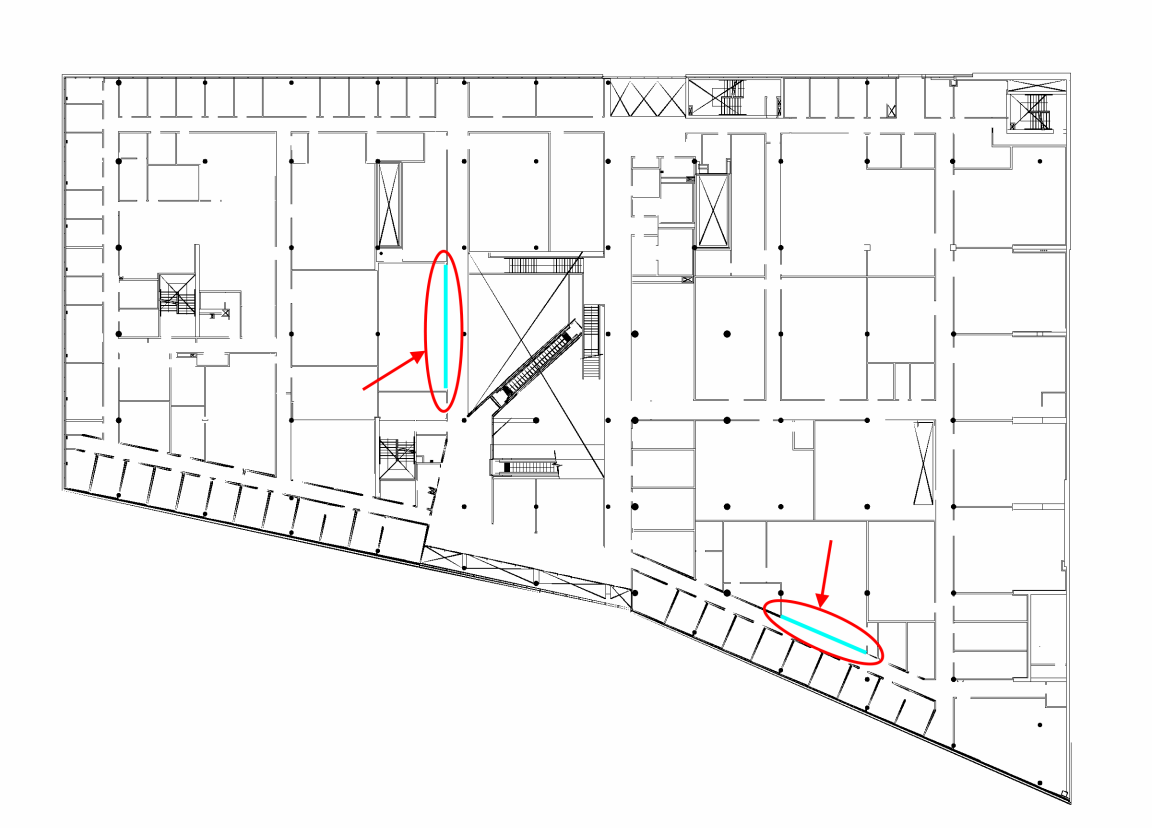

Figure 4.19 Carte construite automatiquement du cinquième étage où deux obstacles manquants sont indiqués manuellement

Nous nous proposons alors d'utiliser notre module d'enrichissement de carte présenté au chapitre précédent pour compléter ces deux obstacles, c'est-à-dire les construire directement en temps réel sur la carte générée de l'étage 5 du pavillon Lassonde.
On active donc les sonars, et on publie notre carte non enrichie sur le réseau du système. Nous avons ainsi tous les inputs nécessaires sauf un seul. Il s'agit de la position du FRMI dans le repère de la carte. Celle-ci est normalement procurée par le composant de localisation du *GMapping*. Vu que dans notre cas actuel la carte est déjà disponible et il ne s'agit pas de la construire avec le *GMapping*, il faut trouver une alternative au composant de localisation du module de SLAM. Cette alternative n'est autre que le module d'auto-localisation.

Les captures montrant l'enrichissement en temps réel pour les deux obstacles sont à la figure 4.20. Notons que les deux obstacles n'ont pas la même transparence.

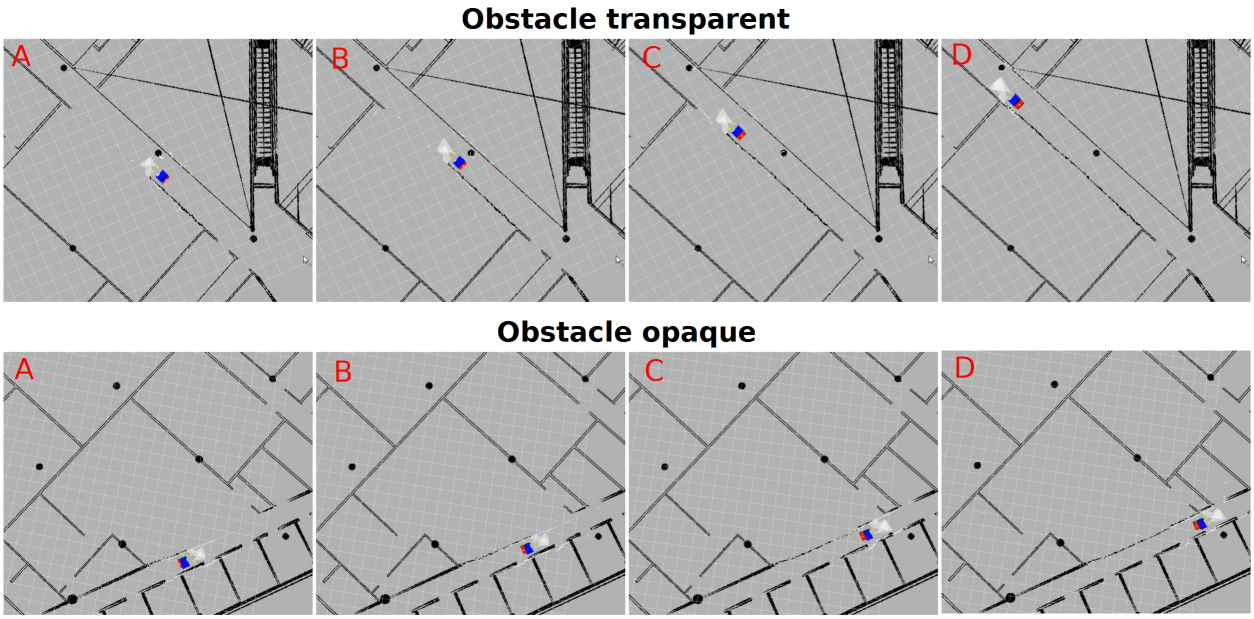

Figure 4.20 Carptures montrant le FRMI en train d'enrichir la carte de l'étage 5 : dessin d'un obstacle transparent et d'un obstacle opaque

En effet, le premier est en vitre, ce qui correspond au cas « naturel » pour lequel le module d'enrichissement a été conçu. Le deuxième est un obstacle opaque. Il est facile de le vérifier sur les captures des deux expériences en regardant les traces des détections laser (en blanc) : ces traces se limitent à des points, représentant les montants qui encadrent les vitres de la façade à dessiner, dans le cas de l'obstacle transparent alors qu'elles montrent une ligne continue dans le cas de l'autre obstacle.

Nous voyons clairement que notre module d'enrichissement est aussi efficace en représentation de vitres qu'en représentation des obstacles opaques. Cela s'explique par le mécanisme de détection qui est le même pour les deux types d'obstacles par rapport aux télémètres ultrason.

La carte complète obtenue à l'issue des opérations d'enrichissement (Figure 4.21) illustre le bon fonctionnement du module sur la carte construite automatiquement. On voit bien que l'obstacle est dessiné correctement au bon emplacement reliant les autres obstacles juxtaposés.

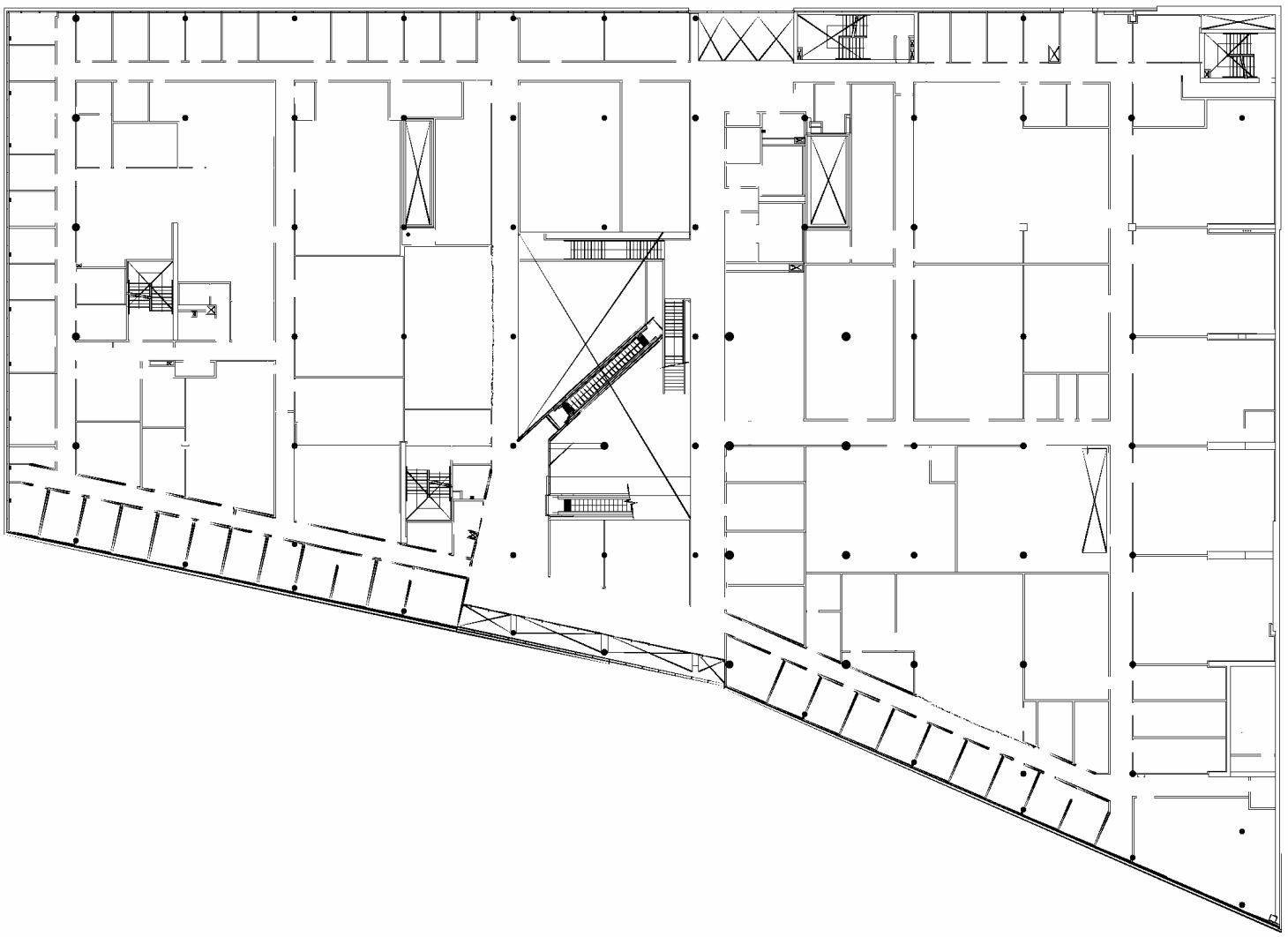

Figure 4.21 Carte construite automatiquement du cinquième étage, enrichie par le module d'enrichissement de carte par données sonar, pour compléter les deux obstacles manquants

4.5 Conclusion

Une brève comparaison globale entre notre module de génération automatique présenté dans ce chapitre et *Le GMapping* utilisé sur le FRMI illustre les avantages et les limites de la solution proposée.

Les avantages de notre module sont alors :

- La rapidité de la construction. La génération automatique est nettement plus rapide que la construction par *le GMapping*.
- L'efficacité supérieure pour la gestion du problème de correspondance. Le problème de correspondance est, en fait, contourné car tous les cycles de l'environnement sont déjà fermés sur le plan d'architecture qui sert de support à la construction.
- Une alternative plus économique. Grâce à notre module, on évite l'exploration de l'environnement qui est généralement coûteuse. En effet, on économise la charge, la durée de vie de la batterie, la durée de vie des capteurs et des joints du robot, l'énérgie de l'opérateur humain, etc.

Par contre, en comparaison au *GMapping*, notre module présente les limites suivantes :

- Il faut obligatoirement avoir un plan à disposition. La solution n'est donc pas fonctionnelle dans certains environnements par opposition au *GMapping* qui fonctionne pour la majorité des milieux.
- La génération automatique est fortement dépendante de la qualité du plan.
- Les cartes générées automatiquement ne représentent généralement pas certains obstacles qu'on ne met pas sur les plans architecturaux (poubelles, armoires, tables, chaises, etc.)

CHAPITRE 5

CONCLUSION

Ce mémoire a permis de présenter deux nouvelles techniques d'optimisation de la modélisation des environnements des robots. Ces deux techniques enrichissent alors le domaine de recherche de la robotique mobile par des solutions testées et prometteuses d'après les résultats exposés dans les chapitres de ce travail.

D'autre part, les solutions ont été implémentées dans un contexte particulier de « robots civiles » où notre robot est un fauteuil roulant motorisé intelligent. Cette dimension de notre recherche montre concrètement l'utilité des résultats obtenus pour le domaine qui vise à apporter plus de confort et de facilité à la vie des personnes à mobilité réduite.

5.1 Synthèse des travaux

Selon les deux grands thèmes de ce mémoire, les réalisations de ce projet sont essentiellement :

Enrichissement de cartes par les obstacles transparents

- La conception d'un module d'enrichissement de carte par données sonars qui permet de compléter efficacement par les obstacles transparents que les télémètres laser ne détectent pas les cartes issues de l'algorithme de *GMapping*.
- Le choix d'une méthode de calcul probabiliste qui permet de minimiser le bruit dans les cartes enrichies tout en assurant une construction consistante et complète des obstacles manquants.
- Un enrichissement continue en temps réel de la carte en cours de construction par le *GMapping* sans conflit avec ce dernier grâce à la priorisation des détections des télémètres laser (car les télémètres laser sont plus performants).
- L'enrichissement est intelligent de façon à profiter des mesures laser prises au-delà des obstacles transparents pour dessiner correctement les zones de la carte situé derrière ces obstacles.
- Le module d'enrichissement de carte est indépendent du module classique de cartographie et il est du coup utilisable en parallèle avec un module d'autolocalisation dans le but d'enrichir une carte statique donnée.

Génération automatique de grilles d'occupation à partir des plans d'architecture

- Établissement et implémentation d'une solution de construction automatique de cartes à partir de plans d'architectures.
- Le format des plans choisi est le format DXF puisque, d'une part, presque tous les autres formats de fichiers peuvent être convertis au format DXF et, d'autre part, la structure vectorielle du plan au format DXF qui est librement consultable facilite énormément l'extraction des éléments nécessaires à la construction.
- La construction automatique de la carte est correcte et précise d'après les résultats obtenus.
- La phase de prétraitement est portable et indépendante des ressources du FRMI.
- La solution de construction automatique offre une alternative extrêmement rapide et nettement plus économique que les algorithmes classiques de cartographie basée sur l'exploration de l'environnement.
- La solution de construction automatique contourne le problème complexe de correspondance rencontré par les algorithmes classiques de SLAM.
- La grille d'occupation obtenue par construction automatique est de dimensions précises et elle est parfaitement compatible avec le système de navigation du FRMI. Nous nous sommes même amusés à la compléter avec succès par certains obstacles manquants à l'aide de notre module d'enrichissement par les mesures des télémètres ultrason.

5.2 Limitations des solutions proposées

5.2.1 Limitations de la solution d'enrichissement de cartes

Le module d'enrichissement de carte par données sonars retourne bien des cartes enrichies efficacement. Cependant, on pense que les résultats d'enrichissement sont étroitement liés aux télémètres ultrason utilisés. Il va sans dire qu'en cas de changement des sonars le réglage des paramètres du module est indispensable pour optimiser l'enrichissement, notamment les paramètres du modèle de perception. Comme nous n'avons pas eu la possiblité de tester différents télémètres ultrason, nous avons eu à adapter le modèle de perception aux seuls sonars que nous avons. Nos sonars étaient de portée médiocre. Ainsi, dans nos expériences d'enrichissement, nous sommes obligés de rouler à une faible distance de l'obstacle transparent à compléter.

D'autre part, vu le petit nombre de sonars utilisés, nous étions obligés de rouler à une vitesse relativement faible le long des obstacles transparents afin d'éviter de faire plusieurs passages pour assurer une construction complète sur la carte et surtout afin d'éviter de cogner l'obstacle en roulant rapidement à une distance dangereusement faible.

5.2.2 Limitations de la solution de génération automatique de cartes

La génération automatique dépend fortement de la qualité du plan d'architecture utilisé comme entrée. Ce plan peut être de qualité médiocre voire insuffisante pour assurer une bonne construction de la grille d'occupation. Par exemple, dans les cas rares d'une répartition très désordonnée des éléments sur les calques du fichier DXF du plan ou d'une spécification incorrecte des dimensions, les répercusions sur la grille obtenue à l'issue de la construction automatique peuvent être importantes au point de la rendre inutilisable pour le système de navigation. Dans ce cas, l'algorithme de SLAM reste quand même la solution de secours à laquelle il faut avoir recours pour faire la cartographie de l'environnement du FRMI.

Une autre limitation de cette solution a été évoquée quand nous avons examiné les résultats de construction automatique des cartes. Il s'agit de l'absence de certains obstacles dans la carte construite et qui sont généralement, en supposant la bonne répartition des éléments sur les calques, des objets non représentés sur les plans architecturaux (poubelles, pots de décoration, armoires placées contre les murs, toute sorte de meubles, etc.)

5.3 Améliorations futures

Les solutions présentées dans ce mémoire ouvrent des perspectives d'amélioration. Nous citons alors les ouvertures suivantes en guise d'améliorations futures :

- Nous proposons de généraliser le module d'enrichissement de cartes à un module d'enrichissement par les mesures des télémètres laser. Cette nouvelle solution qui profite des grandes performances des lasers constitue une solution efficace et beacoup plus performante au problème d'obstacles manquants dans les cartes générées automatiquement. Bien évidemment, le module d'enrichissement proposé par nos travaux reste la solution à utiliser pour compléter les obstacles transparents.
- Nous proposons également de développer la phase de prétraitement de l'algorithme de construction automatique pour intégrer une interface graphique intuitive permettant à l'utilisateur de choisir interactivement les calques en visualisant simultanément les éléments de chaque calque choisi se rajouter au plan simplifié affiché en temps réel.
- Il serait aussi intéressant de tester l'enrichissement de cartes avec des télémètres ultrason de meilleure qualité, en particulier de meilleure portée. Augmenter le nombre des sonars est également possible et fortement conseillé tout en veillant à vérifier la rapidité du calcul probabiliste et, en cas de besoin, on accélère l'algorithme au niveau du calcul par des techniques algorithmiques tels que l'utilisation des *Lookup tables*.

RÉFÉRENCES

AUTODESK (2010). *DXF Reference*. Autodesk, Inc.

BOUCHER, P. (2010). *Navigation Semi-Autonome d'un Fauteuil Roulant Motorise en Environnements Restreints*. Mémoire de maîtrise, Ecole Polytechnique de Montreal.

CASTELLANOS, J., MONTIEL, J., NEIRA, J. et TARDOS, J. (1999). The spmap : A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5), 948–953.

CHEESEMAN, P. et SMITH, P. (1986). On the representation and estimation of spatial uncertainty. *International Journal of Robotics*, 5, 56–68.

DEMPSTER, A., LAIRD, A. et RUBIN, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, Series B, 39(1), 1–38.

DIOSI, A. et KLEEMAN, L. (2004). Advanced sonar and laser range finder fusion for simultaneous localization and mapping. *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. vol. 2.

DIOSI, A., TAYLOR, G. et KLEEMAN, L. (2005). Interactive SLAM using Laser and Advanced Sonar. *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*.

DISSANAYAKE, G., DURRANT-WHYTE, H. et BAILEY, T. (2000). A computationally efficient solution to the simultaneous localisation and map building (slam) problem. *Working notes of ICRA'2000 Workshop W4 : Mobile Robot Navigation and Mapping*.

DISSANAYAKE, G., NEWMAN, P., CLARK, S., DURRANT-WHYTE, H. et CSORBA, M. (2001). A solution to the simultaneous localisation and map building (slam) problem. IEEE Transactions of Robotics and Automation.

DOUCET, A. (1998). On sequential simulation-based methods for bayesian filtering. Rapport technique, Signal Processing Group, Dept. of Engeneering, University of Cambridge.

DOUCET, A., DE FREITAS, N., MURPHY, K. et RUSSELL, S. (2000). Rao-blackwellized particle filtering for dynamic bayesian networks. *In Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*. 176–183.

DUCKETT, T. et KRAJNIK, T. (2014). A frequency-based approach to long-term robotic mapping. *ICRA 2014 Workshop on Long Term Autonomy, June 1st 2014, Hong Kong*.

DURRANT-WHYTE, H., MAJUMDER, S., THURN, S., DE BATTISTA, M. et SCHEDING, S. (2001). A bayesian algorithm for simultaneous localization and map building. *Proceedings of the 10th International Symposium of Robotics Research (ISRR '01)*.

EL-FATHI, A. (2012). *Navigation globale d'un fauteuil roulant motorise dans de grands espaces interieurs*. Mémoire de maîtrise, Ecole Polytechnique de Montreal.

ELFES, A. (1987). Sonar-based real-world mapping and navigation. *IEEE JOURNAL OF ROBOTICS AND AUTOMATION*, RA-3, NO. 3, 249–265.

ELFES, A. (1989). *Occupancy Grid : A Probabilistic Framework for Robot Perception and Navigation*. Thèse de doctorat, Department of Electrical and Computer Engineering, Carnegie Mellon University.

FARHAN, H. A., OWAIED, H. H. et AL-GHAZI, S. I. (2011). Developing cognitive map using blueprint map. *Trends in Applied Sciences Research 6*, 8, 848–862.

FOX, D. (2001). KLD-sampling : Adaptive particle filters and mobile robot localization. Rapport technique, Department of Computer Science & Engineering University of Washington.

GAMBINO, F., ORIOLO, G. et ULIVI, G. (1996). A comparison of three uncertainty calculus techniques for ultrasonic map building.

GRISETTI, G., STACHNISS, C. et BURGARD, W. (2005). Improving grid-based SLAM with rao-blackwellized particle filters by adaptive proposals and selective resampling. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2432–2437.

GRISETTI, G., STACHNISS, C. et BURGARD, W. (2006). Improving techniques for grid mapping with rao-blackwellized particle filters. *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*.

GRISETTI, G., STACHNISS, C., GRZONKA, S. et BURGARD, W. (2007). A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. *Proceedings of Robotics : Science and Systems (RSS)*.

KAESS, M., RANGANATHAN, A. et DELLAERT, F. (2007). iSAM : Fast Incremental Smoothing and Mapping with Efficient Data Association. *Robotics and Automation, 2007 IEEE International Conference on*.

KAESS, M., RANGANATHAN, A. et DELLAERT, F. (2008). iSAM : Incremental Smoothing and Mapping. *Robotics, IEEE Transactions on*. vol. 24.

KALMAN, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, No. 82 (Series D), 35–45.

KRAJNIK, T., FENTANES, J. P., CIELNIAK, G., DONDRUP, C. et DUCKETT, T. (2014). Spectral analysis for long-term robotic mapping. *2014 IEEE International Conference on Robotics and Automation (ICRA 2014), May 31 - June 7, 2014, Hong Kong, China*.

LEONARD, J. et FEDER, H. (1999). A computationally efficient method for large-scale concurrent mapping and localization. J. Hollerbach et D. Koditschek, éditeurs, *Proceedings of the Ninth International Symposium on Robotics Research*.

MONTEMERLO, M. (2003). *FastSLAM : A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. Thèse de doctorat, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

MONTEMERLO, M., WHITTAKER, W. et THURN, S. (2002). Conditional particle filters for simultaneous mobile robot localization and people-tracking. *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

MORAVEC, H. (1988). Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2), 61–74.

MORAVEC, H. P. et ELFES, A. (1984). High resolution maps from wide angle sonar.

MOUTARLIER, P. et CHATILA, R. (1989a). An experimental system for incremental environment modeling by an autonomous mobile robot. *1st International Symposium on Experimental Robotics*.

MOUTARLIER, P. et CHATILA, R. (1989b). Stochastic multisensory data fusion for mobile robot location and environment modeling. *5th Int. Symposium on Robotics Research*.

MURPHY, K. (1999). Bayesian map learning in dynamic environments. *In Proc. of the Conf. on Neural Information Processing Systems (NIPS)*. 1015–1021.

NEWMAN, P. (2000). *On the Structure and Solution of the Simultaneous Localisation and Map Building Problem*. Thèse de doctorat, Australian Center for Field Robotics, University of Sydney, Sydney, Australia.

O'KANE, J. M. (2013). *A Gentle Introduction to ROS*. CreateSpace Independent Publishing Platform.

OLSON, E., LEONARD, J. et TELLER, S. (2006). Fast iterative optimization of pose graphs with poor initial estimates. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

OLSON, E., LEONARD, J. et TELLER, S. (2007). Spatially-adaptive learning rates for online incremental slam. *Proceedings of Robotics : Science and Systems*.

RIBO, M. et PINZ, A. (2001). A comparison of three uncertainty calculi for building sonar-based occupancy grids. *Robotics and Autonomous Systems*, 35, 201–209.

SCHAFER, M., KNAPP, C. et CHAKRABORTY, S. (2011). Automatic generation of topological indoor maps for real-time map-based localization and tracking. *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*.

SMITH, R., SELF, M. et CHEESEMAN, P. (1990). *Autonomous Robot Vehicles*, Springer-Verlag, chapitre Estimating uncertain spatial relationships in robotics. 167–193.

STACHNISS, C., GRISETTI, G. et BURGARD, W. (2005). Recovering particle diversity in a rao-blackwellized particle filter for slam after actively closing loops. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 667–672.

THRUN, S. (2001). A probabilistic online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*, 20(5), 335–363.

THRUN, S. (2002). Robotic mapping : A survey. Rapport technique, School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213.

THRUN, S., FOX, D. et BURGARD, W. (1998). A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31, 29–53.

THRUN, S., WOLFRAM BURGARD et DIETER FOX (2005). *Probabilistic robotics*. Library of Congress Cataloging-in-Publication Data.

TOLMAN, E. (1948). Cognitive maps in rats and men. *Psychological Review*, 55(4), 189–208.

ANNEXE A

Trois méthodes de calcul probabiliste de grilles d'occupation par les mesures des sonars

Nous résumons ici les troix méthodes de calcul probabiliste inspirées des travaux de Ribo et Pinz et Gambino *et al.*[1].

Pour modéliser les sonars, on pose les deux fonctions Γ et Δ qui vont servir à la construction du modèle de perception de chacune des trois méthodes. On suppose que le sonar envoie une onde sonore sous la forme d'un cône d'un demi-angle $h_{\delta W}$ au sommet (on appelle ce cône le cône de détection) . On appelle θ l'angle que fait une cellule de la grille d'occupation à l'axe du cône de détection et ρ sa distance au centre du sonar. Ainsi, les modulations angulaire Γ et radiale Δ s'écrivent :

$$\begin{aligned}\Gamma(\theta) &= \begin{cases} 0; & |\theta| > h_{\delta W} \\ 1 - \dfrac{1}{(h_{\delta W}\pi/180)^2}(\dfrac{\theta\pi}{180})^2; & 0 \leq |\theta| \leq h_{\delta W} \end{cases} \\ \Delta(\rho) &= 1 - \frac{1 + tanh(2(\rho - \rho_\nu))}{2}\end{aligned}$$

où ρ_ν est la distance au-delà de laquelle les mesures du sonar ne sont plus fiables.

Pour ce qui suit, on note r_t la mesure du sonar à l'instant t. Et C_{ij} est une cellule de la grille d'occupation. $C_{ij} = O$ signifie que C_{ij} est occupée et $C_{ij} = L$ signifie que C_{ij} est libre.

A.1 Première méthode : « *Probabilistic approach* »

On se propose de calculer la probabilité d'occupation de chaque cellule de la grille d'occupation. On profite alors de la règle de Bayes pour établir une relation de mise à jour de la probabilité d'occupation de chaque cellule.

$$\begin{aligned}P\ (C_{ij} = O|r_1..r_t) &= \frac{p(r_t|C_{ij} = O)P(C_{ij} = O|r_1..r_{t-1})}{p(r_t|C_{ij} = O)P(C_{ij} = O|r_1..r_{t-1}) + p(r_t|C_{ij} = E)P(C_{ij} = E|r_1..r_{t-1})} \\ &= \frac{p(r_t|C_{ij} = O)P(C_{ij} = O|r_1..r_{t-1})}{p(r_t|C_{ij} = O)P(C_{ij} = O|r_1..r_{t-1}) + [1 - p(r_t|C_{ij} = O)][1 - P(C_{ij} = O|r_1..r_{t-1})]}\end{aligned}$$

1. Nous avons corrigé ici quelques fautes de frappe trouvés dans ces travaux

$P(C_{ij} = O)$ est la valeur a priori de la probabilité d'occupation d'une cellule de la grille avec laquelle on démarre le processus. Cette valeur est prise égale à 0.5.

Dans cette relation, il nous reste à pouvoir calculer le terme $p(r_t|C_{ij} = O)$, c'est à dire le modèle inverse de perception de notre sonar. Le modèle que proposent Ribo et Pinz est (après simplifications dues à nos choix des paramètres de calcul) :

$$p(r|C_{ij} = O) = p_1 + p_2$$

$$p_1 = \begin{cases} \frac{1-\Gamma(\theta)\Delta(\rho)}{2}; & 0 \leq \rho < r - 2\delta r \\ \frac{1-\Gamma(\theta)\Delta(\rho)}{2}(\frac{r-\rho-2\delta r}{\delta r})^2; & r - 2\delta r \leq \rho < r - \delta r \\ \frac{\Gamma(\theta)\Delta(\rho)}{2}(1 - \frac{r-\rho}{\delta r})^2; & r - \delta r \leq \rho < r + \delta r \\ 0; & r + \delta r \leq \rho \end{cases}$$

$$p_2 = \begin{cases} 0; & 0 \leq \rho < r - \delta r \\ 0.5; & r - \delta r \leq \rho \end{cases}$$

δr est une constante à choisir convenablement. Nous proposons de la choisir supérieure à la résolution de la grille d'occupation.

A.2 Deuxième méthode : « *Evidence theoretic approach* »

Soit r une mesure du sonar. Pour cette méthode et celle qui suit nous utilisons un modèle de perception double en introduisant les fonctions :

$$f_L(\rho, r) = \begin{cases} k_L; & 0 \leq \rho < r - \delta r \\ k_L(\frac{r-\rho}{\delta r})^2; & r - \delta r \leq \rho < r \\ 0; & r \leq \rho \end{cases}$$

$$f_O(\rho, r) = \begin{cases} 0; & 0 \leq \rho < r - \delta r \\ k_O[1 - (\frac{r-\rho}{\delta r})^2]; & r - \delta r \leq \rho < r + \delta r \\ 0; & r + \delta r \leq \rho \end{cases}$$

On associe à chaque cellule C_{ij} de la grille une fonction de masse m_{ij} et on se propose de calculer itérativement $m_{ij}(O)$ et $m_{ij}(L)$. La relation de mise à jour de la grille d'occupation

s'écrit alors :

$$\begin{aligned} m_{ij}^{r_1..r_t}(L) &= \frac{m_{ij}^{r_1..r_{t-1}}(L)(1-m_{ij}^{r_t}(O)) + (1-m_{ij}^{r_1..r_{t-1}}(L) - m_{ij}^{r_1..r_{t-1}}(O))m_{ij}^{r_t}(L)}{1-m_{ij}^{r_t}(L)m_{ij}^{r_1..r_{t-1}}(O) - m_{ij}^{r_t}(O)m_{ij}^{r_1..r_{t-1}}(L)} \\ m_{ij}^{r_1..r_t}(O) &= \frac{m_{ij}^{r_1..r_{t-1}}(O)(1-m_{ij}^{r_t}(L)) + (1-m_{ij}^{r_1..r_{t-1}}(L) - m_{ij}^{r_1..r_{t-1}}(O))m_{ij}^{r_t}(O)}{1-m_{ij}^{r_t}(L)m_{ij}^{r_1..r_{t-1}}(O) - m_{ij}^{r_t}(O)m_{ij}^{r_1..r_{t-1}}(L)} \end{aligned}$$

Les valeurs initiales de $m_{ij}(O)$ et $m_{ij}(L)$ sont nulles et à chaque itération $m_{ij}^{r_t}(K)$ où $K = O$ ou $K = L$ est calculé selon la formule $m_{ij}^{r_t}(K) = \Gamma(\theta)\Delta(\rho)f_K(\rho, r_t)$.

L'occupation d'une cellule donnée est finalement concluse par comparaison des deux masses.

A.3 Troixième méthode : « *Possibilistic approach* »

On définit deux fonctions d'appartenance à l'ensemble des cellules libres et à l'ensemble des cellules occupées, respectivement, μ_L et μ_O. On se propose alors ici de quantifier l'appartenance de C_{ij} à ces deux ensembles. On veut donc calculer $\mu_L(C_{ij})$ et $\mu_O(C_{ij})$.

Ainsi, la logique de mise à jour permet d'écrire :

$$\begin{aligned} \mu_L^{r_1..r_t}(C_{ij}) &= \mu_L^{r_1..r_{t-1}}(C_{ij}) + \mu_L^{r_t}(C_{ij}) - \mu_L^{r_1..r_{t-1}}(C_{ij})\mu_L^{r_t}(C_{ij}) \\ \mu_O^{r_1..r_t}(C_{ij}) &= \mu_O^{r_1..r_{t-1}}(C_{ij}) + \mu_O^{r_t}(C_{ij}) - \mu_O^{r_1..r_{t-1}}(C_{ij})\mu_O^{r_t}(C_{ij}) \end{aligned}$$

On initialise les fonctions à 0 ce qui correspond à la non appartenance à aucun des deux ensembles. Et pour calculer $\mu_K^{r_t}(C_{ij})$, on utilise encore le modèle de perception $\Gamma(\theta)\Delta(\rho)f_K(\rho, r_t)$.

Enfin, pour cette méthode la conclusion sur l'occupation des cellules est plus conservative. Elle est déduite par le calcul de l'appartenance à l'ensemble des cellules occupées en intersection avec l'ensemble contenant les cellules qui ne sont ni libres ni occupées, les cellules qui ne sont pas libres et occupées en même temps et les cellules qui ne sont pas libres.